

Projecte de base de dades

PokéSallianWorld 2022-2023 – Fase 3

Llistat de membres (nom i correu):

Joan Tarragó Pina – j.tarrago@students.salle.url.edu

Leonardo Rubén Edenak – leonardoruben.edenak@students.salle.url.edu

Pol Guarch Bosom – pol.guarch@students.salle.url.edu

Andrea Ballester – andrea.ballester@students.salle.url.edu

Data d'entrega: 28/05/23

Taula de continguts

1	INTRODUCCIÓ (1 PÀGINA)	5
2	ACTUALITZACIÓ DEL MODEL ENTITAT-RELACIÓ	6
3	ACTUALITZACIÓ DEL MODEL RELACIONAL	7
4	ACTUALITZACIÓ DEL MODEL FÍSIC	8
5	ELS POKÉMON SÓN GUERRERS ÚNICS	10
5.1	CONSULTA 1	10
5.1.1	Solució.....	10
5.1.2	Explicació	10
5.1.3	Validació	10
5.2	CONSULTA 2	11
5.2.1	Solució.....	11
5.2.2	Explicació	11
5.2.3	Validació	11
5.3	CONSULTA 3	11
5.3.1	Solució.....	11
5.3.2	Explicació	12
5.3.3	Validació	12
5.4	CONSULTA 4	12
5.4.1	Solució.....	12
5.4.2	Explicació	13
5.4.3	Validació	13
5.5	CONSULTA 5	13
5.5.1	Solució.....	13
5.5.2	Explicació	14
5.5.3	Validació	14

5.6	CONSULTA 6	15
5.6.1	Solució.....	15
5.6.2	Explicació	15
5.6.3	Validació	16
5.7	DISPARADOR 1	16
5.7.1	Solució.....	16
5.7.2	Explicació	17
5.7.3	Validació	17
5.8	DISPARADOR 2	17
5.8.1	Solució.....	17
5.8.2	Explicació	19
5.8.3	Validació	19
6	NO SOC UN JUGADOR, SOC UN ENTRENADOR POKÉMON	20
6.1	CONSULTA 1	20
6.1.1	Solució.....	20
6.1.2	Explicació	20
6.1.3	Validació	20
6.2	CONSULTA 2	21
6.2.1	Solució.....	21
6.2.2	Explicació	21
6.2.3	Validació	21
6.3	CONSULTA 3	21
6.3.1	Solució.....	21
6.3.2	Explicació	22
6.3.3	Validació	22
6.4	CONSULTA 4	22
6.4.1	Solució.....	22
6.4.2	Explicació	23
6.4.3	Validació	23
6.5	CONSULTA 5	23
6.5.1	Solució.....	23
6.5.2	Explicació	23
6.5.3	Validació	24
6.6	CONSULTA 6	24
6.6.1	Solució.....	24
6.6.2	Explicació	24
6.6.3	Validació	25
6.7	DISPARADOR 1	26
6.7.1	Solució.....	26
6.7.2	Explicació	27
6.7.3	Validació	27
6.8	DISPARADOR 2	28
6.8.1	Solució.....	28
6.8.2	Explicació	28
6.8.3	Validació	29
7	ENS ANEM DE POKECOMPRES	30
7.1	CONSULTA 1	30
7.1.1	Solució.....	30
7.1.2	Explicació	30
7.1.3	Validació	30
7.2	CONSULTA 2	31
7.2.1	Solució.....	31
7.2.2	Explicació	31
7.2.3	Validació	32
7.3	CONSULTA 3	33
7.3.1	Solució.....	33

7.3.2	Explicació	33
7.3.3	Validació	33
7.4	CONSULTA 4	34
7.4.1	Solució.....	34
7.4.2	Explicació	34
7.4.3	Validació	34
7.5	CONSULTA 5	35
7.5.1	Solució.....	35
7.5.2	Explicació	35
7.5.3	Validació	35
7.6	CONSULTA 6	36
7.6.1	Solució 1.....	36
7.6.2	Solució 2.....	36
7.6.3	Explicació	36
7.6.4	Validació	36
7.7	DISPARADOR 1	37
7.7.1	Solució.....	37
7.7.2	Explicació	37
7.7.3	Validació	38
7.8	DISPARADOR 2	39
7.8.1	Solució.....	39
7.8.2	Explicació	40
7.8.3	Validació	40
8	HORA D'EXPLORAR.....	42
8.1	CONSULTA 1	42
8.1.1	Solució 1.....	42
8.1.2	Solució 2.....	42
8.1.3	Explicació	42
8.1.4	Validació	43
8.2	CONSULTA 2	43
8.2.1	Solució.....	43
8.2.2	Explicació	44
8.2.3	Validació	44
8.3	CONSULTA 3	45
8.3.1	Solució.....	45
8.3.2	Explicació	45
8.3.3	Validació	46
8.4	CONSULTA 4	47
8.4.1	Solució.....	47
8.4.2	Explicació	47
8.4.3	Validació	48
8.5	CONSULTA 5	48
8.5.1	Solució.....	48
8.5.2	Explicació	49
8.5.3	Validació	49
8.6	CONSULTA 6	50
8.6.1	Solució.....	50
8.6.2	Explicació	51
8.6.3	Validació	51
8.7	DISPARADOR 1	52
8.7.1	Solució.....	52
8.7.2	Explicació	53
8.7.3	Validació	54
8.8	DISPARADOR 2	54
8.8.1	Solució.....	54
8.8.2	Explicació	56
8.8.3	Validació	56

9	PREGUNTES CREUADES	58
9.1	CONSULTA 1	58
9.1.1	Solució.....	58
9.1.2	Explicació	58
9.1.3	Validació	58
9.2	CONSULTA 2	59
9.2.1	Solució.....	59
9.2.2	Explicació	60
9.2.3	Validació	60
9.3	CONSULTA 3	61
9.3.1	Solució.....	61
9.3.2	Explicació	61
9.3.3	Validació	61
9.4	CONSULTA 4	62
9.4.1	Solució.....	62
9.4.2	Explicació	63
9.4.3	Validació	63
9.5	CONSULTA 5	63
9.5.1	Solució.....	63
9.5.2	Explicació	64
9.5.3	Validació	64
9.6	CONSULTA 6	65
9.6.1	Solució.....	65
9.6.2	Explicació	66
9.6.3	Validació	66
9.7	CONSULTA 7	67
9.7.1	Solució.....	67
9.7.2	Explicació	68
9.7.3	Validació	69
9.8	CONSULTA 8	69
9.8.1	Solució.....	69
9.8.2	Explicació	70
9.8.3	Validació	70
10	CONCLUSIONS	71
10.1	RECURSOS EMPRATS	71
10.2	US D'IA (SI CAL, 1-2 PÀGINES)	71
10.3	LLIÇONS APRESSES I CONCLUSIONS (1 PÀGINA)	71

1 Introducció (1 pàgina)

Tot el que es comença acaba arribant al final, i es que PokeSallianWorld ja ha estat dissenyat, implementat i fins i tot ja han aparegut les primeres persones i Pokémon. Només falta aplicar la lògica i comprovar que totes les dades es puguin obtenir sense problemes, i tot això, és el que farem a continuació.

En aquesta tercera i última fase del projecte, implementarem una sèrie de consultes que ens permetran obtenir informació del joc. També crearem una sèrie de disparadors que ens permetran poder automatitzar molts dels procediments que tenen lloc a la base de dades que hem creat.

En aquest cas les consultes ens serveixen per acabar d'arreglar petits errors que haguem pogut cometre a l'hora d'implementar la base de dades, a part d'obtenir informació de diverses taules i mostrar-la per pantalla de manera adient. Els disparadors o "triggers" serveixen per donar molt més de realisme a la base de dades, de tal manera que totes les taules quedin unides en certa forma i actualitzades de manera coherent durant les aventures que realitzaran els entrenadors que estan amb ganes de viatjar per tot el món a descobrir noves espècies Pokémon.

L'equip de desenvolupadors que durem a terme aquest projecte serà:

Pol Guach Bosom: S'encarregarà del mòdul Pokémons dels dos models (*Els Pokémons són guerrers únics*)

Joan Tarragó Pina: S'encarregarà del mòdul Entrenadors dels dos models (*No soc un jugador, soc entrenador Pokémon*)

Andrea Ballester Griful: S'encarregarà del mòdul Objectes dels dos models (*Ens n'anem de Pokecompres*)

Leonardo Ruben Edenak Chouev: S'encarregarà del mòdul Exploració dels dos models (*Hora d'explorar*)

2 Actualització del model entitat-relació

No aplica.

3 Actualització del model relacional

No aplica.

4 Actualització del model físic

S'ha afegit un camp que ens vam deixar a la fase 2 a la taula Pokemon_Caught, què és el camp "Gender". Ho hem fet de la següent manera.

```
CREATE TABLE IF NOT EXISTS POKEMON_CAUGHT (
    id_pokemon_caught SERIAL PRIMARY KEY,
    id_trainer INTEGER REFERENCES TRAINER(id_trainer),
    nick_name VARCHAR(50),
    xp INTEGER,
    level INTEGER,
    gender VARCHAR(50),
    id_subarea INTEGER REFERENCES SUBAREA(id_subarea),
    date_caught DATE,
    pokeball_used_caught VARCHAR(50),
    obtaining_method VARCHAR(50),
    pokemon_status_condition VARCHAR(50),
    pokemon_max_HP INTEGER,
    pokemon_remaining_HP INTEGER,
    id_pokemon INTEGER REFERENCES POKEMON(id_pokemon),
    id_nature INTEGER REFERENCES NATURE(id_nature),
    id_pokemon_ability INTEGER REFERENCES ABILITY(id_ability),
    id_object INTEGER REFERENCES ITEM(id_object),
    id_move1 INTEGER REFERENCES POKEMON_MOVE(id_move),
    id_move2 INTEGER REFERENCES POKEMON_MOVE(id_move),
    id_move3 INTEGER REFERENCES POKEMON_MOVE(id_move),
    id_move4 INTEGER REFERENCES POKEMON_MOVE(id_move)
);

INSERT INTO POKEMON_CAUGHT (
    id_pokemon_caught, id_subarea, id_trainer, nick_name, xp, level,
    date_caught, obtaining_method, id_pokemon, pokeball_used_caught,
    pokemon_max_hp, pokemon_status_condition, pokemon_remaining_hp,
    id_move1, id_move2, id_move3, id_move4, id_nature,
    id_pokemon_ability, id_object, gender)
SELECT DISTINCT
    PII.id, SUBAREA.id_subarea, ownerID, nickname, experience,
    level, datetime, obtention_method, pokemon_speciesID, pokeballID,
    INSERTION_TO_TEAMS.hp, INSERTION_TO_TEAMS.status,
    (SELECT max(INSERTION_BATTLE_STATS.remaining_hp)
     FROM INSERTION_BATTLE_STATS
     where INSERTION_BATTLE_STATS.pokemon_instanceID = PII.id),
    M1.id_move, M2.id_move, M3.id_move, M4.id_move, NE.id_nature,
    (SELECT id_ability FROM ABILITY_POKEMON
     JOIN POKEMON ON ABILITY_POKEMON.id_pokemon = POKEMON.id_pokemon
     WHERE POKEMON.id_pokemon = PII.pokemon_speciesID
     ORDER BY RANDOM()
     LIMIT 1),
    IM.id_object, PII.gender
FROM
    POKEMON_INSTANCES_INSERTION AS PII
LEFT JOIN SUBAREA ON SUBAREA.id_subarea = PII.location_subareaID
LEFT JOIN INSERTION_TO_TEAMS ON INSERTION_TO_TEAMS.pokemon_number =
PII.id
INNER JOIN POKEMON_MOVE AS M1 ON M1.id_move = PII.move1
LEFT JOIN POKEMON_MOVE AS M2 ON M2.id_move = PII.move2
LEFT JOIN POKEMON_MOVE AS M3 ON M3.id_move = PII.move3
LEFT JOIN POKEMON_MOVE AS M4 ON M4.id_move = PII.move4
LEFT JOIN ITEM AS IM ON IM.name_object = PII.item
INNER JOIN NATURE AS NE ON NE.name_nature = PII.nature
;
```


S'ha creat un nova Taula anomenada WARNING_TABLE que conté la següent informació:

```
CREATE TABLE IF NOT EXISTS WARNING_TABLE (  
    id SERIAL PRIMARY KEY,  
    id_user VARCHAR(100),  
    id_trainer INTEGER,  
    table_affected VARCHAR (100),  
    warning_message VARCHAR (1000),  
    date_time date  
);
```

S'ha creat com a nou requeriment d'aquesta fase, per tant, no és error nostre la no creació d'aquesta taula. Però ho posem aquí, perquè sense aquesta taula no funcionaran al menys 4 dels disparadors que s'han fet.

5 Els Pokémon són guerrers únics

5.1 Consulta 1

5.1.1 Solució

```
SELECT POKEMON.name_pokemon as nom, POKEMON.weight_pokemon as pes,  
stat1.base_value as defensa, stat2.base_value as defensa_especial  
FROM ABILITY_POKEMON  
INNER JOIN POKEMON ON  
POKEMON.id_pokemon = ABILITY_POKEMON.id_pokemon  
INNER JOIN ABILITY ON  
ABILITY.id_ability = ABILITY_POKEMON.id_ability  
INNER JOIN ACQUIRE_BASE_STAT as stat1 ON  
stat1.id_pokemon = POKEMON.id_pokemon  
INNER JOIN BASE_STAT as stat_defense ON  
stat_defense.id_base_stat = stat1.id_base_stat  
INNER JOIN ACQUIRE_BASE_STAT as stat2 ON  
stat2.id_pokemon = POKEMON.id_pokemon  
INNER JOIN BASE_STAT as stat_defense_sp ON  
stat_defense_sp.id_base_stat = stat2.id_base_stat  
WHERE ABILITY.name_ability LIKE 'sturdy' AND stat_defense.name  
LIKE 'defense' AND stat_defense_sp.name LIKE 'special defense'  
ORDER BY pes DESC  
LIMIT 10;
```

nom character varying (50)	pes integer	defensa integer	defensa_especial integer
steelix	4000	200	65
aggron	3600	180	60
probopass	3400	145	150
golem	3000	130	65
regirock	2300	200	100
onix	2100	160	45
magnezone	1800	115	90
bastiodon	1495	168	138
fortress	1258	140	60
dorphan	1200	120	60

5.1.2 Explicació

Aquesta consulta selecciona el nom del Pokémon, el seu pes, el valor de la seva defensa i el valor de la seva defensa especial. Es realitza una sèrie d'unions (INNER JOIN) entre diverses taules per obtenir aquestes dades. Les unions es realitzen entre les taules ABILITY_POKEMON i POKEMON basant-se en els identificadors de Pokémon, entre les taules ABILITY i ABILITY_POKEMON basant-se en els identificadors d'habilitats, i entre les taules ACQUIRE_BASE_STAT i POKEMON i entre BASE_STAT i POKEMON per obtenir les estadístiques de defensa i defensa especial. Es realitza una clàusula WHERE per filtrar els resultats només per a les habilitats amb el nom "sturdy" i les estadístiques de defensa i defensa especial amb els noms corresponents. Els resultats es ordenen per pes en ordre descendent i es limiten a les 10 primeres files..

5.1.3 Validació

La consulta està ben construïda i lògicament estructurada. Utilitza instruccions "JOIN" per combinar adequadament les taules relacionades. La clàusula "WHERE" filtra els Pokémon amb l'habilitat "sturdy" i amb l'estadística de defensa i defensa especial corresponents. La consulta

selecciona el nom del Pokémon, el pes, el valor de la defensa i el valor de la defensa especial. A més, la consulta ordena els resultats per pes de manera descendent i limita el resultat a 10 entrades.

5.2 Consulta 2

5.2.1 Solució

```
SELECT AVG(stat_attack.base_value) as mitjana_attack,  
MIN(stat_attack.base_value) as  
min_attack, MAX(stat_attack.base_value) as max_attack FROM POKEMON  
INNER JOIN TYPE_POKEMON as type1 ON  
type1.id_type = POKEMON.id_type_1  
INNER JOIN TYPE_POKEMON as type2 ON  
type2.id_type = POKEMON.id_type_2  
INNER JOIN ACQUIRE_BASE_STAT as stat_attack ON  
stat_attack.id_pokemon = POKEMON.id_pokemon  
INNER JOIN BASE_STAT as stat ON  
stat.id_base_stat = stat_attack.id_base_stat  
WHERE (type1.name_type LIKE 'fire' OR type2.name_type LIKE 'fire')  
AND stat.name LIKE 'attack';
```

mitjana_attack	min_attack	max_attack
numeric	integer	integer
88.5384615384615385	50	130

5.2.2 Explicació

La consulta calcula la mitjana, el valor mínim i el valor màxim de les estadístiques d'atac dels Pokémon de tipus "fire". Es realitzen unions (INNER JOIN) entre diverses taules per obtenir les dades necessàries. Es realitzen unions entre les taules POKEMON i TYPE_POKEMON per relacionar els Pokémon amb els seus tipus, i es realitzen unions entre les taules ACQUIRE_BASE_STAT i POKEMON i entre BASE_STAT i ACQUIRE_BASE_STAT per obtenir les estadístiques d'atac. Es fa servir una clàusula WHERE per filtrar els resultats només pels Pokémon amb tipus "fire" i les estadístiques amb el nom attack. Finalment, es calcula la mitjana, el valor mínim i el valor màxim de les estadístiques d'atac utilitzant les funcions AVG, MIN i MAX respectivament.

5.2.3 Validació

La consulta està ben estructurada i construïda de manera lògica. Utilitza instruccions "JOIN" per combinar taules relacionades de manera adequada. La clàusula "WHERE" filtra els Pokémon amb tipus "foc" en el tipus 1 o tipus 2, i també filtra aquells Pokémon amb l'estadística "atac". La consulta utilitza funcions d'agregació com AVG, MIN i MAX per calcular la mitjana, el valor mínim i el valor màxim de l'atribut d'atac dels Pokémon seleccionats.

5.3 Consulta 3

5.3.1 Solució

```
SELECT POKEMON.name_pokemon , ABILITY.name_ability,  
ABILITY.full_description FROM POKEMON  
INNER JOIN ABILITY_POKEMON ON  
ABILITY_POKEMON.id_pokemon = POKEMON.id_pokemon  
INNER JOIN ABILITY ON  
ABILITY_POKEMON.id_ability = ABILITY.id_ability  
WHERE POKEMON.base_xp_pokemon > (  
SELECT AVG(POKEMON.base_xp_pokemon) FROM POKEMON
```

)
ORDER BY ABILITY.name_ability, POKEMON.name_pokemon DESC;

name_pokemon character varying (50)	name_ability character varying (50)	full_description character varying (5000)
porygon-z	adaptability	This Pokémon inflicts twice as much damage with moves whose types match its own, rather than the usual same-type attack bonus of 1.5×.
crawdaunt	adaptability	This Pokémon inflicts twice as much damage with moves whose types match its own, rather than the usual same-type attack bonus of 1.5×.
skuntank	aftermath	When this Pokémon is knocked out by a move that makes contact, the move's user takes 1/4 its maximum HP in damage.
electrode	aftermath	When this Pokémon is knocked out by a move that makes contact, the move's user takes 1/4 its maximum HP in damage.
drifblim	aftermath	When this Pokémon is knocked out by a move that makes contact, the move's user takes 1/4 its maximum HP in damage.
rayquaza	air lock	While this Pokémon is in battle, weather can still be in play, but will not have any of its effects. This ability functions identically to cloud nine.
starmie	analytic	This Pokémon's moves have 1.3× their power when it moves last in a turn. future sight and doom desire are unaffected.
porygon2	analytic	This Pokémon's moves have 1.3× their power when it moves last in a turn. future sight and doom desire are unaffected.
porygon-z	analytic	This Pokémon's moves have 1.3× their power when it moves last in a turn. future sight and doom desire are unaffected.
magnezone	analytic	This Pokémon's moves have 1.3× their power when it moves last in a turn. future sight and doom desire are unaffected.
magnetron	analytic	This Pokémon's moves have 1.3× their power when it moves last in a turn. future sight and doom desire are unaffected.
tauros	anger point	Whenever this Pokémon receives a critical hit, its Attack rises to the maximum of 6 stages. This ability will still take effect if the critical hit is received by a substitute.
primeape	anger point	Whenever this Pokémon receives a critical hit, its Attack rises to the maximum of 6 stages. This ability will still take effect if the critical hit is received by a substitute.
camerupt	anger point	Whenever this Pokémon receives a critical hit, its Attack rises to the maximum of 6 stages. This ability will still take effect if the critical hit is received by a substitute.
wormadam-plant	anticipation	When this Pokémon enters battle, if one of its opponents has a move that is super effective against it, self destruct, explosion, or a one-hit knockout move, all participati
whiscash	anticipation	When this Pokémon enters battle, if one of its opponents has a move that is super effective against it, self destruct, explosion, or a one-hit knockout move, all participati

(Hi ha 595 files en total)

5.3.2 Explicació

Aquesta consulta, selecciona el nom del Pokémon, el nom de l'habilitat i la descripció completa de l'habilitat per als Pokémon que tenen una experiència de base més gran que la mitjana de tots els Pokémon. Es realitzen unions (INNER JOIN) entre diverses taules per obtenir les dades requerides. Es realitza una subconsulta per calcular la mitjana de l'experiència de base de tots els Pokémon i es fa servir aquest valor per filtrar els resultats. Els resultats s'ordenen per nom de l'habilitat en ordre ascendent i per nom del Pokémon en ordre descendent.

5.3.3 Validació

La consulta està ben estructurada i construïda de manera lògica. Utilitza una combinació d'instruccions "JOIN" per unir les taules relacionades de manera adequada. A més, la subconsulta dins de la clàusula "WHERE" calcula la mitjana de l'atribut base_xp_pokemon a la taula POKEMON i compara el resultat amb el base_xp_pokemon de cada Pokémon a la taula principal per filtrar els resultats. La consulta també ordena els resultats pel nom de l'habilitat en ordre ascendent i, en cas d'empat, pel nom del Pokémon en ordre descendent.

5.4 Consulta 4

5.4.1 Solució

```
SELECT base_stat.name, post.name_pokemon, stat.base_value FROM BASE_STAT AS
base_stat
INNER JOIN ACQUIRE_BASE_STAT AS stat ON
stat.id_base_stat = base_stat.id_base_stat
INNER JOIN POKEMON AS baby ON stat.id_pokemon = baby.id_pokemon
INNER JOIN EVOLVES ON
EVOLVES.id_pokemon_base = baby.id_pokemon
INNER JOIN POKEMON AS post ON
EVOLVES.id_pokemon_final = post.id_pokemon
WHERE EVOLVES.baby = TRUE AND stat.base_value = (
    SELECT MAX(maxstat.base_value) FROM
    ACQUIRE_BASE_STAT as maxstat
    INNER JOIN BASE_STAT as current_stat ON current_stat.id_base_stat =
    maxstat.id_base_stat
    INNER JOIN POKEMON as pokemon_baby ON maxstat.id_pokemon =
    pokemon_baby.id_pokemon
    INNER JOIN EVOLVES ON EVOLVES.id_pokemon_base = pokemon_baby.id_pokemon
    INNER JOIN POKEMON AS post ON EVOLVES.id_pokemon_final = post.id_pokemon
    WHERE EVOLVES.baby = TRUE AND
base_stat.name = current_stat.name
```

);

name character varying (20)	name_pokemon character varying (50)	base_value integer
attack	snorlax	85
defense	sudowoodo	95
hp	snorlax	135
special attack	jynx	85
special defense	mantine	120
speed	electabuzz	95

5.4.2 Explicació

L'anterior consulta, obté el nom de l'estadística de base, el nom del Pokémon evolucionat i el valor de l'estadística de base per als Pokémon bebès que tenen la estadística de base màxima en comú amb la seva forma evolucionada. Es realitzen unions (INNER JOIN) entre diverses taules per obtenir les dades requerides. Es realitza una subconsulta per obtenir el valor màxim de l'estadística de base comuna entre els bebès i les seves formes evolucionades. Aquest valor es fa servir per filtrar els resultats. S'aplica una clàusula WHERE per assegurar que només s'obtenen els Pokémon bebès i es compara la estadística de base amb el valor màxim obtingut a la subconsulta.

5.4.3 Validació

La consulta està ben construïda i lògicament estructurada. Utilitza instruccions "JOIN" per combinar adequadament les taules relacionades. La consulta busca les estadístiques màximes d'un Pokémon que evoluciona, filtrant els resultats segons si és un Pokémon inicial o bebè. També s'assegura que la estadística màxima coincideixi amb la mateixa estadística del Pokémon bebè. La consulta selecciona el nom de l'estadística, el nom del Pokémon post-evolució i el valor de la estadística.

```
SELECT base_stat.name, post.name_pokemon, stat.base_value FROM
BASE_STAT AS base_stat
INNER JOIN ACQUIRE_BASE_STAT AS stat ON
stat.id_base_stat = base_stat.id_base_stat
INNER JOIN POKEMON AS baby ON stat.id_pokemon = baby.id_pokemon
INNER JOIN EVOLVES ON EVOLVES.id_pokemon_base = baby.id_pokemon
INNER JOIN POKEMON AS post ON
EVOLVES.id_pokemon_final = post.id_pokemon
WHERE EVOLVES.baby = TRUE AND base_stat.name LIKE '%attack'
ORDER BY stat.base_value DESC;
```

5.5 Consulta 5

5.5.1 Solució

```
(SELECT tipus.name_type as mes_fortalesa, 'MES FORT' as n FROM
TYPE_POKEMON as tipus
INNER JOIN TYPE_AGAINST as contra ON
contra.id_type_attacker = tipus.id_type
WHERE contra.multiplier > 1
GROUP BY tipus.name_type
HAVING COUNT(contra.multiplier) = (
SELECT COUNT(TYPE_AGAINST.multiplier) FROM TYPE_AGAINST
INNER JOIN TYPE_POKEMON ON
TYPE_POKEMON.id_type = TYPE_AGAINST.id_type_attacker
WHERE TYPE_AGAINST.multiplier > 1
```

```

GROUP BY TYPE_POKEMON.name_type
ORDER BY COUNT(TYPE_AGAINST.multiplier) DESC
LIMIT 1)
LIMIT 1)

```

UNION

```

(SELECT tipus.name_type as mes_fluix, 'MES FLUIX' as n FROM
TYPE_POKEMON as tipus
INNER JOIN TYPE_AGAINST as contra ON
contra.id_type_defender = tipus.id_type
WHERE contra.multiplier < 1
GROUP BY tipus.name_type
HAVING COUNT(contra.multiplier) = (
SELECT COUNT(TYPE_AGAINST.multiplier) FROM TYPE_AGAINST
INNER JOIN TYPE_POKEMON ON
TYPE_POKEMON.id_type = TYPE_AGAINST.id_type_defender
WHERE TYPE_AGAINST.multiplier < 1
GROUP BY TYPE_POKEMON.name_type
ORDER BY COUNT(TYPE_AGAINST.multiplier) DESC
LIMIT 1)
LIMIT 1)

```

mes_fortalesa	n
character varying (50)	text
ground	MES FORT
steel	MES FLUIX

5.5.2 Explicació

Aquesta cinquena consulta, obté el tipus de Pokémon que té la major fortalesa (més efectiu) en la primera subconsulta i el tipus de Pokémon que té la major debilitat (menys efectiu) en la segona subconsulta. Es realitzen unions (UNION) entre les dues subconsultes. Cada subconsulta realitza unions (INNER JOIN) entre les taules TYPE_POKEMON i TYPE_AGAINST per obtenir les dades necessàries. Es realitzen filtres per trobar els tipus amb un multiplicador més gran que 1 (subconsulta de major fortalesa) o amb un multiplicador menor que 1 (subconsulta de major debilitat). Es fa servir una clàusula GROUP BY per agrupar els resultats segons el nom del tipus i es fa servir una clàusula HAVING per assegurar que només s'obtenen els tipus amb el mateix nombre de multiplicadors que el tipus amb més multiplicadors (per major fortalesa) o que el tipus amb més multiplicadors negatius (per major debilitat). Es realitza una subconsulta addicional per obtenir el nombre màxim de multiplicadors o multiplicadors negatius, i es fa servir una clàusula ORDER BY i LIMIT per obtenir el tipus amb aquest nombre màxim. El resultat final és una combinació (UNION) del tipus de major fortalesa i el tipus de major debilitat amb les seves respectives etiquetes ('MES FORT' i 'MES FLUIX').

5.5.3 Validació

La consulta està ben construïda i lògicament estructurada. Utilitza les instruccions "JOIN" per combinar adequadament les taules relacionades. La consulta cerca el tipus de Pokémon amb la fortalesa més gran i el tipus de Pokémon amb la debilitat més gran. Utilitza una subconsulta per trobar el nombre de vegades que cada tipus de Pokémon té una fortalesa o debilitat superior o inferior a 1. A continuació, selecciona el tipus de Pokémon amb el nombre més gran de fortalese o debilitats i els etiqueta com a "MES FORT" i "MES FLUIX", respectivament.

5.6 Consulta 6

5.6.1 Solució

```
WITH RECURSIVE c AS (  
    SELECT POK.id_pokemon AS id, id_pokemon AS inicial ,  
    'Cad: ' || id_pokemon AS Cadena, GR.xp AS level_up  
        FROM POKEMON as POK  
        JOIN GROWTH_RATE AS GR ON GR.id_pk = POK.id_growth_rate  
        JOIN EVOLVES AS EV ON POK.id_pokemon =  
EV.id_pokemon_base  
    UNION ALL  
    SELECT EV.id_pokemon_final, EV.id_pokemon_base, c.cadena || '-  
' || EV.id_pokemon_final, GR.xp+c.level_up  
        FROM POKEMON as POK  
        JOIN GROWTH_RATE AS GR ON GR.id_pk = POK.id_growth_rate  
        JOIN EVOLVES AS EV ON POK.id_pokemon =  
EV.id_pokemon_final  
        JOIN c ON c.id = EV.id_pokemon_base  
  
)  
SELECT * --c.id,c.inicial, max(c.cadena), max(level_up)  
FROM c  
WHERE c.level_up =  
(select max(c.level_up) FROM c  
WHERE c.id <> c.inicial)  
;
```

5.6.2 Explicació

Aquesta consulta utilitza una clàusula "WITH RECURSIVE" per construir una cadena recursiva de Pokémon que evolucionen entre ells fins a arribar a la pujada de nivell màxima.

La consulta està dividida en dues parts que s'uniran mitjançant l'operador UNION ALL. Aquesta és la part recursiva de la consulta.

En la primera part, s'obtenen els Pokémon inicials i les seves respectives pujades de nivell a partir de les taules "POKEMON", "GROWTH_RATE" i "EVOLVES". S'utilitzen les condicions de join adequades per relacionar aquestes taules i seleccionar el camp "id_pokemon" com a identificador de cada Pokémon. Aquesta part retorna els següents camps: "id" (identificador del Pokémon), "inicial" (identificador del Pokémon inicial), "Cadena" (cadena inicial que inclou l'identificador del Pokémon) i "level_up" (pujada de nivell del Pokémon).

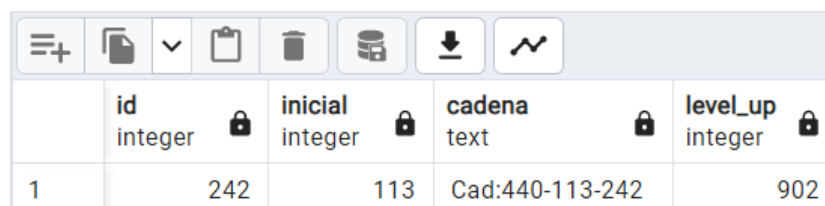
En la segona part, s'obtenen les evolucions següents dels Pokémon inicials. Aquesta part utilitza les mateixes taules i condicions de join que la part anterior. També s'uneix a la consulta recursiva "c" mitjançant la condició "JOIN c ON c.id = EV.id_pokemon_base". Això permet referir-se als registres anteriors de la consulta recursiva i afegir-los a la cadena de la evolució actual. A més, s'incrementa el valor de "level_up" sumant-li el camp "level_up" del Pokémon actual. Aquesta part retorna els mateixos camps que la primera part.

Després de la part recursiva, s'executa la part final de la consulta que selecciona tots els camps de la consulta recursiva "c" on el "level_up" sigui igual al valor màxim de "level_up" trobat a la consulta recursiva "c" (excluint els registres de Pokémon inicial). Això ens donarà la cadena de Pokémon que arriba a la pujada de nivell màxima.

Com a resultat, la consulta retorna els camps "id" (identificador del Pokémon), "inicial" (identificador del Pokémon inicial), "Cadena" (cadena completa de la evolució) i "level_up" (pujada de nivell corresponent a la cadena completa de la evolució).

5.6.3 Validació

Fotografia del resultat:



	id integer	inicial integer	cadena text	level_up integer
1	242	113	Cad:440-113-242	902

Podem veure com la cadena evolutiva sembla que sigui la de Happini, Chansey i Blissey que requereixen de 902 punts d'experiència per arribar a la seva etapa evolutiva final.

5.7 Disparador 1

5.7.1 Solució

```
CREATE OR REPLACE FUNCTION check_more_than_limit()
RETURNS TRIGGER AS $$
BEGIN
    -- Busquem que existeixi el moviment que se li passi en la
    motxilla de l'entrenador
    IF EXISTS (
        SELECT 1
        FROM ABILITY_POKEMON AS AB
        WHERE AB.id_pokemon = 34
        AND AB.is_hidden = false
        GROUP BY AB.id_pokemon
        HAVING count(AB.id_pokemon) > 2
    ) THEN
        INSERT INTO WARNING_TABLE (id_user, date_time,
warning_message)
        SELECT current_user, current_timestamp,
        format('A %s the entry has been inserted into the
ABILITY POKEMON table', count(AB.id_pokemon))
        FROM ABILITY_POKEMON AS AB
        WHERE AB.id_pokemon = new.id_pokemon AND AB.is_hidden =
false
        GROUP BY AB.id_pokemon
        ;

    RETURN NULL;
```



```

        END IF;
        RETURN NEW;
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS check_abilities ON ABILITY_POKEMON;
CREATE TRIGGER check_abilities
AFTER INSERT ON ABILITY_POKEMON
FOR EACH ROW
EXECUTE FUNCTION check_more_than_limit();

```

5.7.2 Explicació

Aquest disparador té com a objectiu controlar el nombre de registres que es poden inserir a la taula "ABILITY_POKEMON" per a un determinat Pokémon. La funció `check_more_than_limit()` s'executa com a disparador després d'una inserció a la taula ABILITY_POKEMON. Aquesta funció comprova si hi ha més de dos registres per al mateix Pokémon (`id_pokemon = 34`) en què l'habilitat no sigui oculta (`is_hidden = false`). Si es compleix aquesta condició, s'insereix un registre a la taula "WARNING_TABLE" amb informació sobre l'avís generat. En cas contrari, es permet la inserció del nou registre a la taula "ABILITY_POKEMON". Això permet controlar i limitar el nombre de registres que es poden afegir per a un Pokémon específic en la taula ABILITY_POKEMON.

5.7.3 Validació

A continuació, unes quantes consultes per poder validar aquest trigger, i també està l'insert que s'ha utilitzat per provar el correcte funcionament d'aquest.

```

SELECT * FROM POKEMON
WHERE id_pokemon = 34;

SELECT * FROM ABILITY_POKEMON
WHERE id_pokemon = 34;

INSERT INTO ABILITY_POKEMON (id_ability, id_pokemon, slot,
is_hidden) VALUES (35, 34, 4, false);

DELETE FROM ABILITY_POKEMON WHERE id_pokemon = 34 and id_ability =
35;

SELECT * FROM WARNING_TABLE;
DELETE FROM WARNING_TABLE;

```

5.8 Disparador 2

5.8.1 Solució

```

CREATE OR REPLACE FUNCTION evolves()
RETURNS TRIGGER AS $$
DECLARE
    possible_evolution INT := 0;
    id_evolution INT := -1;
BEGIN
    IF EXISTS (
        SELECT ES.min_level
        FROM POKEMON AS POK
        JOIN EVOLVES AS ES ON POK.id_pokemon = ES.id_pokemon_base
    )

```

```

        WHERE new.level >= ES.min_level AND ES.method_initial = 'level up' AND
        POK.id_pokemon = new.id_pokemon
    ) THEN
        IF EXISTS (
            SELECT 1
            FROM POKEMON AS POK
            JOIN EVOLVES AS ES ON POK.id_pokemon = ES.id_pokemon_base
            WHERE (ES.gender = '' OR ES.gender IS NULL) AND (ES.time_of_day = '' OR
            ES.time_of_day IS NULL) AND POK.id_pokemon = 387 -- new.id_pokemon
        ) THEN

            possible_evolution := 1;
        END IF;

        IF EXISTS (
            SELECT 1
            FROM POKEMON AS POK
            JOIN EVOLVES AS ES ON POK.id_pokemon = ES.id_pokemon_base
            WHERE ES.gender = 'female' AND ES.gender = new.gender
            AND POK.id_pokemon = new.id_pokemon
        ) THEN
            possible_evolution := 2;
        END IF;

        IF EXISTS (
            SELECT 1
            FROM POKEMON AS POK
            JOIN EVOLVES AS ES ON POK.id_pokemon = ES.id_pokemon_base
            WHERE ES.gender = 'male' AND ES.gender = new.gender
            AND POK.id_pokemon = new.id_pokemon
        ) THEN
            possible_evolution := 2;
        END IF;

        IF EXISTS (
            SELECT 1
            FROM POKEMON AS POK
            JOIN EVOLVES AS ES ON POK.id_pokemon = ES.id_pokemon_base
            WHERE ES.time_of_day = 'night' AND POK.id_pokemon = new.id_pokemon
            AND (
                (date_part('hour', current_time) >= 20 AND date_part('hour',
current_time) <= 24)
                OR (date_part('hour', current_time) >= 0 AND date_part('hour',
current_time) < 8)
            )
        ) THEN
            possible_evolution := 1;
        END IF;

        IF EXISTS (
            SELECT 1
            FROM POKEMON AS POK
            JOIN EVOLVES AS ES ON POK.id_pokemon = ES.id_pokemon_base
            WHERE ES.time_of_day = 'night' AND POK.id_pokemon = new.id_pokemon
            AND (date_part('hour', current_time) < 20 AND date_part('hour',
current_time) >= 8)
        ) THEN
            possible_evolution := 1;
        END IF;

        IF possible_evolution = 1 THEN
            SELECT ES.id_pokemon_final INTO id_evolution
            FROM POKEMON AS POK
            JOIN EVOLVES AS ES ON POK.id_pokemon = ES.id_pokemon_base
            WHERE POK.id_pokemon = new.id_pokemon;

            UPDATE POKEMON_CAUGHT SET id_pokemon = id_evolution
            WHERE POKEMON_CAUGHT.id_pokemon_caught = new.id_pokemon_caught;
        END IF;

```

```

        IF possible_evolution = 2 THEN
        SELECT ES.id_pokemon_final INTO id_evolution
        FROM POKEMON AS POK
        JOIN EVOLVES AS ES ON POK.id_pokemon = ES.id_pokemon_base
            WHERE POK.id_pokemon = new.id_pokemon
            AND EV.gender = new.gender;

            INSERT INTO WARNING_TABLE (id_trainer) VALUES (id_evolution);
            new.id_pokemon := id_evolution;
        END IF;
    ELSE
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS evolution ON POKEMON_CAUGHT;
CREATE TRIGGER evolution
AFTER UPDATE ON POKEMON_CAUGHT
FOR EACH ROW
WHEN (NEW.level <> OLD.level)
EXECUTE FUNCTION evolves()
;

```

5.8.2 Explicació

Aquest disparador es dispara després d'una actualització a la taula "POKEMON_CAUGHT". La funció "evolves()" s'executa per a cada fila actualitzada i comprova diverses condicions per determinar si hi ha una possible evolució del Pokémon capturat. Aquesta funció fa servir una sèrie de consultes per verificar les condicions de l'evolució, com ara el nivell mínim, el gènere i el moment del dia. Si es compleixen aquestes condicions, es realitzen diferents accions, com actualitzar el Pokémon capturat amb la seva nova forma o inserir un registre a la taula "WARNING_TABLE". Si no es compleixen les condicions d'evolució, no es realitza cap acció. Aquest disparador permet gestionar l'evolució dels Pokémon capturats i prendre les accions adequades en funció de les condicions especificades.

5.8.3 Validació

A continuació, unes quantes consultes per poder validar aquest trigger, i també està l'update que s'ha utilitzat per provar el correcte funcionament d'aquest.

```

SELECT POK.name_pokemon, PC.* FROM POKEMON_CAUGHT AS PC
JOIN POKEMON AS POK ON POK.id_pokemon = PC.id_pokemon
JOIN EVOLVES AS EV ON EV.id_pokemon_base = POK.id_pokemon;

```

```

SELECT * FROM POKEMON_CAUGHT AS PC
WHERE PC.id_pokemon_caught = 8;

```

```

SELECT * FROM EVOLVES
WHERE id_pokemon_base = 387;

```

```

UPDATE POKEMON_CAUGHT SET level = 20
WHERE id_pokemon_caught = 8;

```

```

SELECT id_pokemon FROM POKEMON_CAUGHT AS PC
WHERE PC.id_pokemon_caught = 8;

```

```

UPDATE POKEMON_CAUGHT SET level = 4, id_pokemon = 387
WHERE id_pokemon_caught = 8;

```

6 No soc un jugador, soc un entrenador Pokémon

6.1 Consulta 1

6.1.1 Solució

```
SELECT PC.nick_name AS Pokemon_Name, PC.id_pokemon_caught AS Id_Pokemon_Caught
FROM POKEMON_CAUGHT AS PC
JOIN TEAM ON TEAM.id_trainer =
(SELECT TR1.id_trainer FROM TRAINER AS TR1
JOIN BATTLE AS BE ON BE.trainer_winner = TR1.id_trainer
JOIN TRAINER AS TR2 ON TR2.id_trainer = BE.trainer_loser
WHERE TR2.item_gift_leader IS NOT null
GROUP BY TR1.id_trainer
ORDER BY count(TR1.name_trainer) DESC
LIMIT 1)
WHERE TEAM.slot = 1 AND PC.id_pokemon_caught = TEAM.id_pokemon_caught
;
```

6.1.2 Explicació

En aquesta query el que es fa és mostrar el sobrenom del Pokémon i l'identificador d'aquest a partir de que estigui en el equip d'aquell "Trainer" que ha de ser el que més "Trainers" amb el distintiu "item_gift_leader is not null" que vol dir que són líders de gimnàs, ha derrotat (per això el límit 1). Després mostrar el que tingui el "slot = 1" de l'equip.

6.1.3 Validació

Resultat:

	pokemon_name character varying (50)	id_pokemon_caught [PK] integer
1	Trouble	1214

En aquesta imatge es pot veure com el Pokémon que surt com el primer de l'equip que més líders de gimnàs ha derrotat és en Trouble.

	id_trainer [PK] integer
1	46

En aquesta imatge surt el resultat d'executar la consulta de validació, que ens mostra a l'entrenador que més vegades ha derrotat a líders de gimnàs. I si li trèiem el LIMIT 1, veiem com surten més i no només és un entrenador el que ha guanyat als líders de gimnàs.

```
SELECT TR1.id_trainer FROM TRAINER AS TR1
JOIN BATTLE AS BE ON BE.trainer_winner = TR1.id_trainer
JOIN TRAINER AS TR2 ON TR2.id_trainer = BE.trainer_loser
WHERE TR2.item_gift_leader IS NOT null
GROUP BY TR1.id_trainer
ORDER BY count(TR1.name_trainer) DESC
--LIMIT 1
;
```

6.2 Consulta 2

6.2.1 Solució

```
SELECT PC.id_pokemon_caught AS pokemon_ID, PC.nick_name AS nickname,  
count(PC.id_pokemon_caught) AS times_defeated, (SELECT max(PC2.date_caught) AS  
date_caughted FROM POKEMON_CAUGHT AS PC2 WHERE PC2.id_trainer = TR1.id_trainer)  
FROM POKEMON_CAUGHT AS PC  
JOIN BATTLE_STATS AS BS ON PC.id_pokemon_caught = BS.id_pokemon_caught  
JOIN BATTLE AS BE ON BE.id_battle = BS.id_battle  
JOIN TRAINER AS TR1 ON TR1.id_trainer = BE.trainer_winner  
WHERE PC.pokemon_max_hp - BS.damage_received = 0 AND PC.id_trainer =  
TR1.id_trainer  
GROUP BY PC.id_pokemon_caught, PC.nick_name, TR1.id_trainer  
ORDER BY 3 DESC, 4  
LIMIT 5  
;
```

6.2.2 Explicació

Mostro l'identificador dels Pokémons, el seu sobrenom i les vegades que han guanyat en combats Pokémon. Però comprovo aquells Pokémon que hagin estat debilitats en aquell combat en la clàusula WHERE. Faig un GROUP BY per poder contar quants cops em surten els Pokémons seleccionats en qüestió. I finalment, ordeno per la columna número 3 de manera descendent per mostrar als 5 Pokémon que més cops han guanyat tot i essent debilitats en combat.

6.2.3 Validació

Aquesta consulta funciona ja que el resultat que es mostra per pantalla coincideix amb les consultes que s'està demanant a la base de dades sobre tot de que el entrenador hagi guanyat el combat però que en aquell combat el Pokémon en qüestió va quedar debilitat.

	pokemon_id integer	nickname character varying (50)	times_defeated bigint	date_caughted date
1	3575	Mac	5	2022-10-25
2	4940	Rhett	4	2021-03-22
3	4939	Nibbles	4	2021-03-22
4	4937	Toffee	4	2021-03-22
5	8111	Ginger	4	2021-05-01

6.3 Consulta 3

6.3.1 Solució

```
SELECT TR1.id_trainer AS trainer, count(PC.id_pokemon_caught) AS  
pokemon_caughted  
FROM TRAINER AS TR1  
JOIN POKEMON_CAUGHT AS PC ON PC.id_trainer = TR1.id_trainer  
JOIN BATTLE AS BE ON TR1.id_trainer <> BE.trainer_winner  
WHERE TR1.exp_trainer > power(TR1.pokeCoins_trainer, 2)  
GROUP BY TR1.id_trainer  
;
```

6.3.2 Explicació

El que es fa en aquesta consulta no té gaire misteri, és ajuntar la taula entrenador amb Pokémon_Caught per fer la recol·lecte del nombre de Pokémon, en el cas favorable de no haver guanyat cap combat, per tant “id.trainer” no ha d’estar en “trainer_winner”, però tot i així tenen més experiència que or elevat al quadrat.

6.3.3 Validació

EL resultat de la consulta és:

	trainer integer	pokemon_caughted bigint
1	396	16254
2	438	9632
3	314	21035
4	426	6030
5	322	6020
6	115	25886
7	108	7826
Total rows: 51 of 51		Query complete 00:00:00.189

Per acabar-lo de validar podem fer la mateixa consulta però seleccionant a més a més altres camps de la base de dades, com ara l’or elevat al quadrat i la experiència.

```
SELECT TR1.id_trainer AS trainer, count(PC.id_pokemon_caught) AS
pokemon_caughted , exp_trainer, power(TR1.pokeCoins_trainer,2)
FROM TRAINER AS TR1
JOIN POKEMON_CAUGHT AS PC ON PC.id_trainer = TR1.id_trainer
JOIN BATTLE AS BE ON TR1.id_trainer <> BE.trainer_winner
WHERE TR1.exp_trainer > power(TR1.pokeCoins_trainer, 2)
GROUP BY TR1.id_trainer
;
```

	trainer integer	pokemon_caughted bigint	exp_trainer integer	power double precision
1	396	16254	1534745	70756
2	438	9632	1612034	366025
3	314	21035	1136959	153664
4	426	6030	365665	94864
5	322	6020	1849052	938961
6	115	25886	728284	577600
7	108	7826	1998634	329476
Total rows: 51 of 51		Query complete 00:00:00.188		

6.4 Consulta 4

6.4.1 Solució

```
SELECT DISTINCT ON (ORG.name_organization) TR1.name_trainer, TR2.name_trainer,
ORG.name_organization,
MAX(ET1.money_stolen + ET2.money_stolen) AS total_money_stolen
FROM EVIL_TRAINER AS ET1
JOIN ORGANIZATION AS ORG ON ET1.id_organization = ORG.id_organization
JOIN TRAINER AS TR1 ON TR1.id_trainer = ET1.id_trainer
JOIN TRAINER AS TR2 ON TR2.id_trainer = ET1.id_partner
```

```
JOIN EVIL_TRAINER AS ET2 ON ET2.id_trainer = ET1.id_partner
GROUP BY ORG.name_organization, TR1.name_trainer, TR2.name_trainer
ORDER BY ORG.name_organization, total_money_stolen DESC
```

;

6.4.2 Explicació

Aquesta consulta utilitza les taules ORGANIZATION, TRAINER i EVIL_TRAINER per poder calcular la quantitat d'or que ha recollit un entrenador malèvol al llarg de la seva carrera. Però com també se li ha de sumar l'or recaptat per la seva parella, es fa ús d'una segona taula TRAINER per poder calcular també aquella quantitat. Finalment quan es mostra el resultat, només es mostra els entrenadors malèvols que hagin tingut la màxima recollida entre tots dos, per cada organització existent a la base de dades.

6.4.3 Validació

El resultat de la consulta és:

	name_trainer character varying (50) 🔒	name_trainer character varying (50) 🔒	name_organization character varying (50) 🔒	total_money_stolen integer 🔒
1	Mia	Hermine	Aqua	2998
2	Carlton	Lincoln	Cipher	1848
3	Terry	Ailene	Galactic	2538
4	Jacinto	Kerri	Magma	2697
5	Oren	Fidel	Rocket	2268

6.5 Consulta 5

6.5.1 Solució

```
SELECT PC1.nick_name, PC1.id_pokemon_caught, PC1.pokemon_status_condition,
PC2.nick_name, PC2.id_pokemon_caught
FROM POKEMON_CAUGHT AS PC1
JOIN BATTLE_STATS AS BS ON PC1.id_pokemon_caught = BS.id_pokemon_caught
JOIN BATTLE AS BE ON BE.id_battle = BS.id_battle
JOIN TRAINER AS TR ON TR.id_trainer = BE.trainer_winner OR TR.id_trainer =
BE.trainer_loser
JOIN TEAM AS TE ON TE.id_trainer = TR.id_trainer
JOIN POKEMON_CAUGHT AS PC2 ON TE.id_pokemon_caught = PC2.id_pokemon_caught
JOIN POKEMON_MOVE AS PM1 ON PM1.id_move = PC2.id_move1
JOIN POKEMON_MOVE AS PM2 ON PM2.id_move = PC2.id_move2
JOIN POKEMON_MOVE AS PM3 ON PM3.id_move = PC2.id_move3
JOIN POKEMON_MOVE AS PM4 ON PM4.id_move = PC2.id_move4
WHERE PC1.pokemon_status_condition <> 'none'
AND (PM1.ailment is not null OR PM2.ailment is not null OR PM3.ailment is
not null OR PM4.ailment is not null)
;
```

6.5.2 Explicació

Aquesta consulta fa molts JOINS (Productes Cartesians) perquè necessitem comprovar cadascun dels moviments d'un Pokémon que s'ha barallat amb un altre Pokémon (que tingui condició d'estat desfavorable), per comprovar si li ha pogut causar la condició d'estat desfavorable amb un dels seus moviments.

6.5.3 Validació

El resultat de la consulta:

	pokemon_with_status_condition character varying (50)	id_pokemon_caught [PK] integer	pokemon_status_condition character varying (50)	pokemon_attacker character varying (50)	id_pokemon_caught integer
1	Jazz	7524	burn	Abigail	0
2	Shorty	1	freeze	Abigail	0
3	Haley	8841	confusion	Abigail	0
4	Shorty	1	freeze	Abigail	0
5	Jazz	7524	burn	Shorty	1
6	Shorty	1	freeze	Shorty	1
7	Haley	8841	confusion	Shorty	1
8	Shorty	1	freeze	Shorty	1
9	Jazz	7524	burn	Greta	2
10	Shorty	1	freeze	Greta	2
11	Haley	8841	confusion	Greta	2
12	Shorty	1	freeze	Greta	2
Total rows: 1000 of 5524 Query complete 00:00:00.092					

Podem veure en aquesta imatge que el Pokémon Jazz amb l'identificador 7524 té diverses possibilitats de Pokémon que li puguin haver causat aquella condició, d'estat.

La consulta que es mostra a continuació la he posat com a exemple de moviment amb probabilitat de provocar una condició d'estat desfavorable. En aquest cas la “cremada” (o “burn” en anglès).

```
SELECT * FROM POKEMON_MOVE WHERE name_move = 'flamethrower';
```

	name_move	flinch_chance integer	min_hits integer	max_hits integer	ailment character varying (50)	ailment_chance integer	status_move character varying (50)	stat_change_rate integer	change_amount integer
1	flamethrower	90	0	1	burn	10	[null]	0	[null]

6.6 Consulta 6

6.6.1 Solució

```
SELECT PT.name_type, count(BE.trainer_loser)
FROM BATTLE AS BE
JOIN TRAINER AS TR1 ON TR1.id_trainer = BE.trainer_loser
JOIN GYM ON GYM.id_trainer = TR1.id_trainer
JOIN TYPE_POKEMON AS PT ON GYM.id_type = PT.id_type
GROUP BY PT.name_type
ORDER BY count(BE.trainer_loser) ASC
;
```

6.6.2 Explicació

Aquesta consulta es basa en que un entrenador líder de gimnàs utilitza un tipus en concret de Pokémon en el seu gimnàs, si aquest és derrotat en algun enfrontament contra algun altre entrenador (qualsevol tipus d'entrenador conta fins i tot altres líders de gimnàs o entrenadors malèvols) sumarà 1 al comptador de entrenadors que han perdut. I com es fa un GROUP BY pel tipus de Pokémon

utilitzat en el gimnàs, sortirà tots els entrenadors d'aquell tipus que han perdut enfrontaments contra altres entrenadors.

6.6.3 Validació

El resultat de la consulta:

	name_type character varying (50) 🔒	count bigint 🔒
1	water	1
2	normal	1
3	fighting	1
4	flying	1
5	ground	1
6	fire	1
7	rock	1
8	electric	2
9	steel	3
10	bug	3
11	ghost	3
12	poison	4

En aquesta consulta no hi ha possibilitat d'error, ja que, l'únic que es fa és contar per cada tipus existent de líders de gimnàs, quants han estat derrotats menys vegades.

6.7 Disparador 1

6.7.1 Solució

Cal crear la taula WARNING_TABLE.

```
DROP TABLE IF EXISTS WARNING_TABLE;
CREATE TABLE IF NOT EXISTS WARNING_TABLE (
    id SERIAL PRIMARY KEY,
    id_user VARCHAR(100),
    id_trainer INTEGER,
    table_affected VARCHAR (100),
    warning_message VARCHAR (1000),
    date_time date
);

CREATE OR REPLACE FUNCTION teach_move_trigger()
RETURNS TRIGGER AS $$
BEGIN

    IF NOT EXISTS (
        SELECT 1
        FROM POKEMON_MOVE AS PM
        JOIN TM_HM AS TH ON TH.id_move = PM.id_move
        JOIN ITEM AS IM ON IM.id_object = TH.id_mt
        JOIN CONTAINS_ITEM AS CI ON CI.id_object = IM.id_object
        JOIN BACKPACK AS BP ON BP.id_backpack = CI.id_backpack
        JOIN TRAINER AS TR ON TR.id_trainer = BP.id_trainer
        WHERE new.id_trainer = TR.id_trainer
            and ( (TH.id_move = new.id_move1 and old.id_move1 <> new.id_move1)
                or (TH.id_move = new.id_move2 and old.id_move2 <> new.id_move2)
                or (TH.id_move = new.id_move3 and old.id_move3 <> new.id_move3)
                or (TH.id_move = new.id_move4 and old.id_move4 <> new.id_move4)
            )
    ) THEN

        INSERT INTO WARNING_TABLE (id_trainer, warning_message)
        SELECT new.id_trainer,
            format('%s %s attempted to teach his %s the move %s without the necessary move machine.',
TR.class_trainer, TR.name_trainer, new.id_pokemon, PM.name_move)
        FROM TRAINER AS TR, POKEMON_MOVE AS PM
        WHERE TR.id_trainer = new.id_trainer
        AND ( (PM.id_move = new.id_move1 and old.id_move1 <> new.id_move1)
            or (PM.id_move = new.id_move2 and old.id_move2 <> new.id_move2)
            or (PM.id_move = new.id_move3 and old.id_move3 <> new.id_move3)
            or (PM.id_move = new.id_move4 and old.id_move4 <> new.id_move4)
        );

        RETURN NULL;
    END IF;

    DELETE FROM CONTAINS_ITEM WHERE (id_backpack, id_object) = (SELECT CI.id_backpack, CI.id_object
        FROM POKEMON_MOVE AS PM
        JOIN TM_HM AS TH ON TH.id_move = PM.id_move
        JOIN ITEM AS IM ON IM.id_object = TH.id_mt
        JOIN CONTAINS_ITEM AS CI ON CI.id_object = IM.id_object
        JOIN BACKPACK AS BP ON BP.id_backpack = CI.id_backpack
        JOIN TRAINER AS TR ON TR.id_trainer = BP.id_trainer
        WHERE TR.id_trainer = new.id_trainer
        AND ( (PM.id_move = new.id_move1 and old.id_move1 <> new.id_move1)
            or (PM.id_move = new.id_move2 and old.id_move2 <> new.id_move2)
            or (PM.id_move = new.id_move3 and old.id_move3 <> new.id_move3)
            or (PM.id_move = new.id_move4 and old.id_move4 <> new.id_move4)
        ));

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS teach_move ON POKEMON_CAUGHT;
CREATE TRIGGER teach_move
AFTER UPDATE
ON POKEMON_CAUGHT
FOR EACH ROW
```

```

WHEN (old.id_move1 is distinct from new.id_move1 or
      old.id_move2 is distinct from new.id_move2 or
      old.id_move3 is distinct from new.id_move3 or
      old.id_move4 is distinct from new.id_move4)
EXECUTE FUNCTION teach_move_trigger();

```

6.7.2 Explicació

Aquest disparador consisteix en trobar si aquell moviment que un entrenador ha ensenyat a un Pokémon (en qualsevol de les posicions de moviment del Pokémon) mitjançant una Màquina Tècnica, està dins de la seva motxilla. Si ho està el que farem serà eliminar el camp de la Màquina Tècnica de dins de la motxilla. I si no l'ha trobat el que farem serà mostrar un missatge d'error a la Taula "warnings_table" conforme aquell entrenador està utilitzant mètodes il·legals per ensenyar aquell moviment al Pokémon, per en un futur denunciar-lo a les autoritats administradores del programa. Com podrem observar en el codi que hi ha a continuació el disparador va acompanyat d'una funció anomenada teach_move_trigger() que té tota la lògica del trigger. Per a què aquesta funció funcioni correctament les taules que es necessiten per les consultes són:

- POKEMON_MOVE
- TM_HM
- CONTAINS_ITEM
- BACKPACK
- TRAINER
- WARNING_TABLE

A més a més de la taula POKEMON_CAUGHT que és on està situada el disparador que executa la funció després de fer un "Update" a aquesta taula (a qualsevol fila) dels camps id_move#1..4.

El que fa la funció és molt senzill, primer busca que existeixi la Màquina Tècnica dins la motxilla del jugador utilitzant un condicional, i es comprova per lògica negativa, és a dir, "IF NOT EXISTS" inserirà en la taula WARNING_TABLE un error i desfarà el canvi del "Update" fet amb un "RETURN NULL". Si, sí que existeix, esborrarà el camp de la Màquina Tècnica que pertoca i es farà el "Update".

6.7.3 Validació

A continuació, unes quantes consultes per poder validar aquest trigger, i també està l'update que s'ha utilitzat per provar el correcte funcionament d'aquest.

```

SELECT * FROM WARNING_TABLE;

SELECT * FROM POKEMON_MOVE WHERE id_move = 1;

SELECT * FROM POKEMON_CAUGHT WHERE id_move1 = 6 and id_trainer = 10;
SELECT * FROM POKEMON_CAUGHT WHERE id_trainer = 10;

SELECT I.name_object FROM CONTAINS_ITEM AS CI
JOIN ITEM AS I ON CI.id_object = I.id_object
WHERE id_backpack = 10;

SELECT *
FROM CONTAINS_ITEM WHERE (id_backpack, id_object) = (SELECT CI.id_backpack,
CI.id_object
FROM POKEMON_MOVE AS PM
JOIN TM_HM AS TH ON TH.id_move = PM.id_move
JOIN ITEM AS IM ON IM.id_object = TH.id_mt
JOIN CONTAINS_ITEM AS CI ON CI.id_object = IM.id_object
JOIN BACKPACK AS BP ON BP.id_backpack = CI.id_backpack

```

```

JOIN TRAINER AS TR ON TR.id_trainer = BP.id_trainer
WHERE TR.id_trainer = 10
AND PM.id_move = 1);

```

```

UPDATE POKEMON_CAUGHT SET id_move1 = 1 WHERE id_move1 = 6 and id_trainer = 10
and level = 4;

```

6.8 Disparador 2

6.8.1 Solució

No cal crear cap taula addicional pel funcionament d'aquest disparador.

```

CREATE OR REPLACE FUNCTION give_gold_and_exp()
RETURNS TRIGGER AS $$
BEGIN
    IF EXISTS (SELECT 1
               FROM TRAINER AS TR
               JOIN EVIL_TRAINER AS ET ON ET.id_trainer = TR.id_trainer
               WHERE TR.id_trainer = new.trainer_winner)
    THEN
        UPDATE TRAINER
        SET pokeCoins_trainer = GREATEST(pokeCoins_trainer + new.pokeCoins_battle * 2 / 3,
0),
            exp_trainer = exp_trainer + new.experience_battle * 2 / 3
        WHERE id_trainer = new.trainer_winner;

        UPDATE TRAINER
        SET pokeCoins_trainer = GREATEST(pokeCoins_trainer - new.pokeCoins_battle, 0),
            exp_trainer = exp_trainer + new.experience_battle * 1 / 3
        WHERE id_trainer = new.trainer_loser;
        INSERT INTO WARNING_TABLE (id_trainer) VALUES (new.trainer_winner);
    ELSE
        UPDATE TRAINER
        SET pokeCoins_trainer = GREATEST(pokeCoins_trainer + new.pokeCoins_battle * 2 / 3,
0),
            exp_trainer = exp_trainer + new.experience_battle * 2 / 3
        WHERE id_trainer = new.trainer_winner;

        UPDATE TRAINER
        SET pokeCoins_trainer = GREATEST(pokeCoins_trainer + new.pokeCoins_battle * 1 / 3,
0),
            exp_trainer = exp_trainer + new.experience_battle * 1 / 3
        WHERE id_trainer = new.trainer_loser;
        /*INSERT INTO WARNING_TABLE (warning_message) VALUES (format('pokeC: %s, EXP: %s,
t_win: %s, t_los: %s', new.pokeCoins_battle, new.experience_battle, new.trainer_winner,
new.trainer_loser));*/
        END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS give_gold_exp ON BATTLE;
CREATE TRIGGER give_gold_exp
AFTER INSERT ON BATTLE
FOR EACH ROW
EXECUTE FUNCTION give_gold_and_exp();

```

6.8.2 Explicació

Aquest disparador té com a objectiu actualitzar els entrenadors després que aquests hagin realitzat un enfrontament Pokémon. Depenent de quin tipus siguin seguirà un criteri o un altre. Si l'entrenador contra el que s'enfronten és un entrenador malèvol i guanya aquest, l'entrenador perdedor pagarà de la seva butxaca a l'entrenador malèvol tots els diners que s'havien apostat en el combat, dels quals l'entrenador malèvol es quedarà una 2/3 part (i suposem que la part restant l'enviarà a l'organització criminal). En canvi si l'enfrontament té lloc amb entrenadors no malèvols, el guanyador guanyarà les 2/3 parts dels diners del combat mentre que el perdedor la 1/3 part. En tots dos casos la experiència dels entrenadors guanyadors augmenta en 2/3 parts l'experiència del combat, mentre que els perdedors en 1/3 part.

Per al correcte funcionament del disparador s'utilitzen les següents taules:

- TRAINER
- EVIL_TRAINER

A més a més de la taula utilitzada per fer la inserció BATTLE que és on està situada el disparador.

Per a l'explicació de la funció del disparador, només cal fixar-se en el condicional IF i ELSE que hi ha, per lo demás és només fer "l'update" a la taula TRAINER amb la nova informació. El condicional aquest cop és per lògica positiva i comprova si existeix un Entrenador amb l'identificador del guanyador dins la taula de EVIL_TRAINER.

6.8.3 Validació

A continuació, unes quantes consultes per poder validar aquest trigger, i també està l'insert que s'ha utilitzat per provar el correcte funcionament d'aquest.

```
SELECT * FROM WARNING_TABLE;
```

```
SELECT * FROM BATTLE LIMIT 1;
```

```
UPDATE TRAINER
SET pokeCoins_trainer = 11297,
    exp_trainer = 1175748
WHERE TRAINER.id_trainer = 511;
```

```
UPDATE TRAINER
SET pokeCoins_trainer = 10210,
    exp_trainer = 1365328
WHERE TRAINER.id_trainer = 511;
```

```
SELECT * FROM TRAINER WHERE id_trainer = 511; -- 11297 pokeCoins inicials,
1175748 exp final
SELECT * FROM TRAINER WHERE id_trainer = 343; -- 10210 pokeCoins inicials,
1365328 exp final
```

```
DELETE FROM BATTLE WHERE id_battle >= 600;
```

```
INSERT INTO BATTLE (id_battle, trainer_winner, trainer_loser, pokeCoins_battle,
experience_battle, duration_battle) VALUES ((SELECT id_battle FROM BATTLE ORDER
BY 1 DESC LIMIT 1) + 1, 511, 343, 999, 999, -1)
```

```
SELECT * FROM BATTLE WHERE id_battle = 600;
```

```
/*
UPDATE TRAINER
SET pokeCoins_trainer = GREATEST(pokeCoins_trainer + 999 * 2 / 3, 0),
    exp_trainer = exp_trainer + 999 * 2 / 3
WHERE id_trainer = 511;
*/
```

```
SELECT * FROM TRAINER WHERE id_trainer = 511; -- 11963 pokeCoins finals, 1176414
exp final
SELECT * FROM TRAINER WHERE id_trainer = 343; -- 10543 pokeCoins finals, 1365661
exp final
```

7 Ens anem de Pokecompres

7.1 Consulta 1

7.1.1 Solució

```
SELECT TRAINER.name_trainer, ITEM.name_object FROM BUYS
INNER JOIN ITEM ON BUYS.id_object = ITEM.id_object
INNER JOIN TRAINER ON BUYS.id_trainer = TRAINER.id_trainer
WHERE TRAINER.name_trainer LIKE (
    SELECT TRAINER.name_trainer FROM BUYS
    INNER JOIN TRAINER ON BUYS.id_trainer = TRAINER.id_trainer
    WHERE BUYS.id_trainer = TRAINER.id_trainer
    GROUP BY TRAINER.name_trainer
    ORDER BY SUM (BUYS.cost) DESC
    LIMIT 1)
GROUP BY TRAINER.name_trainer, ITEM.name_object
ORDER BY SUM (BUYS.amount) DESC
LIMIT 1;
```

name_trainer character varying (50)	name_object character varying (50)
Eliseo	blue scarf

7.1.2 Explicació

La utilització d'un LEFT JOIN en la consulta permet incloure totes les files de la taula "BUYS", independentment de si hi ha coincidències amb les taules "ITEM" i "TRAINER". Això assegura que es recuperen totes les compres realitzades. La subconsulta s'utilitza per seleccionar el trainer amb el màxim cost acumulat de compres, utilitzant l'agregació SUM i l'ordenació DESC per obtenir aquesta selecció específica. Finalment, l'ús de la clàusula GROUP BY permet agrupar els resultats per trainer i objecte, permetent un anàlisi agregada i facilitant l'ordenació basada en el total d'unitats comprades. Es limita el resultat a 1 per a que ens mostri només el primer resultat que és el que ens interessa ja que ho hem ordenat.

7.1.3 Validació

Per validar que la consulta està bé, primer (1) he buscat l'entrenador que s'ha gastat més diners a la botiga obtenint com a resultat 'Eliseo'. A partir d'això, (2) he buscat quin és l'objecte que ha comprat més vegades 'Eliseo' ja que era el jugador que s'havia gastat més diners a la botiga. Com a resultat he obtingut el objecte 'blue scarf' que l'ha comprat 100 vegades.

(1)

```
SELECT TRAINER.name_trainer FROM BUYS
INNER JOIN TRAINER ON BUYS.id_trainer = TRAINER.id_trainer
WHERE BUYS.id_trainer = TRAINER.id_trainer
GROUP BY TRAINER.name_trainer
ORDER BY SUM (BUYS.cost) DESC
LIMIT 1;
```

name_trainer character varying (50)
Eliseo

(2)

```
SELECT ITEM.name_object, SUM (BUYS.amount) AS total_bought FROM
BUYS
INNER JOIN ITEM ON BUYS.id_object = ITEM.id_object
INNER JOIN TRAINER ON BUYS.id_trainer = TRAINER.id_trainer
WHERE TRAINER.name_trainer LIKE ('Eliseo')
GROUP BY ITEM.name_object
ORDER BY total_bought DESC
LIMIT 1;
```

name_object character varying (50)	total_bought bigint
blue scarf	100

7.2 Consulta 2

7.2.1 Solució

```
SELECT DISTINCT(TRAINER.name_trainer) FROM TRAINER
WHERE TRAINER.exp_trainer > 3000
AND TRAINER.name_trainer IN (
    SELECT TRAINER.name_trainer FROM TRAINER
    INNER JOIN BACKPACK ON TRAINER.id_trainer =
BACKPACK.id_trainer
    INNER JOIN CONTAINS_ITEM ON CONTAINS_ITEM.id_backpack =
BACKPACK.id_backpack
    INNER JOIN ITEM ON ITEM.id_object = CONTAINS_ITEM.id_object
    GROUP BY TRAINER.name_trainer
    HAVING COUNT(DISTINCT(ITEM.id_object)) > 10
)
AND TRAINER.name_trainer IN (
    SELECT TRAINER.name_trainer FROM TRAINER
    LEFT JOIN BUYS ON TRAINER.id_trainer = BUYS.id_trainer
    WHERE BUYS.id_trainer IS NULL);
```

name_trainer character varying (50)
Abel
Alan
Alda
Alex
Apryl
Archie
Bonita
Brian
Chad
Christopher
Clora
Collette
Corrie
Domonique
...

(n'hi ha més, 62 files en total)

7.2.2 Explicació

S'utilitza un LEFT JOIN per incloure tots els trainers, independentment de si tenen o no compres registrades a la taula "BUYS". Això permet identificar els trainers que no tenen cap compra. Es fa

servir una subconsulta per obtenir els noms dels trainers que tenen més de 10 objectes únics al seu backpack, utilitzant joins amb les taules "BACKPACK", "CONTAINS_ITEM" i "ITEM". A més, s'utilitza una altra subconsulta per seleccionar els trainers amb una experiència superior a 3000. Finalment, s'aplica la clàusula GROUP BY per agrupar els resultats per nom del trainer, garantint noms únics en la sortida.

7.2.3 Validació

Per validar que la consulta està bé, (1) busquem tots els entrenadors que tenen més de 3000 punts d'experiència, (2) llistem els entrenadors que mai han comprat res a la botiga i (3) finalment, cerquem els jugadors que dins la seva motxilla tenen més de 10 objectes diferents.

(1)

```
SELECT TRAINER.name_trainer, TRAINER.exp_trainer FROM TRAINER
WHERE TRAINER.exp_trainer > 3000
ORDER BY TRAINER.name_trainer ASC;
```

name_trainer character varying (50)	exp_trainer integer
Abbie	58526
Abdul	944578
Abel	1632589
Abraham	1675006
Abraham	1328421
Adan	1949245
Adeline	726088
Aida	1843460
Ailene	1445867
Alan	950558

(n'hi ha més, 527 files en total)

(2)

```
SELECT TRAINER.name_trainer, TRAINER.id_trainer FROM TRAINER
LEFT JOIN BUYS ON TRAINER.id_trainer = BUYS.id_trainer
WHERE BUYS.id_trainer IS NULL
ORDER BY TRAINER.name_trainer ASC;
```

name_trainer character varying (50)	id_trainer [PK] integer
Ana	281
Apryl	334
Archie	532
Barton	116
Bella	338
Birgit	32
Bonita	102
Brian	65
Byron	529
Carmen	103
Cecily	391

(n'hi ha més, 102 files en total)

(3)

```
SELECT TRAINER.name_trainer, COUNT(DISTINCT(ITEM.id_object)) AS
objects FROM TRAINER
INNER JOIN BACKPACK ON TRAINER.id_trainer = BACKPACK.id_trainer
INNER JOIN CONTAINS_ITEM ON CONTAINS_ITEM.id_backpack =
BACKPACK.id_backpack
INNER JOIN ITEM ON ITEM.id_object = CONTAINS_ITEM.id_object
GROUP BY TRAINER.name_trainer
```



```
HAVING COUNT(DISTINCT(ITEM.id_object)) > 10
ORDER BY TRAINER.name_trainer ASC;
```

name_trainer character varying (50)	objects bigint
Abdul	17
Abel	11
Abraham	27
Adan	16
Adeline	29
Aida	28
Ailene	22
Alan	31
Albertina	13
Alda	26
Aldo	12
Alex	26
Andy	12
Angelika	31
Angelita	11

(n'hi ha més, 332 files en total)

7.3 Consulta 3

7.3.1 Solució

```
SELECT FLAVOUR.name FROM ITEM
JOIN BERRY ON BERRY.id_berry = ITEM.id_object
JOIN SELLS ON SELLS.id_object = ITEM.id_object
JOIN BERRY_FLAVOUR ON BERRY.id_berry = BERRY_FLAVOUR.id_berry
JOIN FLAVOUR ON FLAVOUR.id_flavour = BERRY_FLAVOUR.id_flavour
GROUP BY FLAVOUR.name
ORDER BY COUNT(FLAVOUR.id_flavour) ASC
LIMIT 1;
```

name character varying (50)
spicy

7.3.2 Explicació

La consulta selecciona el nom del sabor de la fruita que té la menor quantitat d'associacions amb altres registres. Es realitzen joins amb les taules "BERRY", "SELLS", "BERRY_FLAVOUR" i "FLAVOUR" per obtenir les dades necessàries. La clàusula GROUP BY s'utilitza per agrupar els resultats pel nom del sabor, mentre que l'ordre es determina utilitzant l'agregació COUNT de les associacions amb el sabor (FLAVOUR.id_flavour). Finalment, s'aplica la clàusula LIMIT per retornar només el primer registre amb la quantitat més petita d'associacions que serà la que menys vegades existeix i per tant, el sabor més difícil de trobar.

7.3.3 Validació

Per validar que la consulta està bé (1) mirem quins són els possibles sabors que poden tenir les berries (objectes que tenen sabor). (2) Contem el nombre d'associacions entre els sabors de fruites i els noms dels sabors corresponents.

(1)

```
SELECT FLAVOUR.name from FLAVOUR;
```

name
character varying (50)
dry
sour
spicy
sweet
bitter

(2)

```
SELECT COUNT(BERRY_FLAVOUR.id_flavour), FLAVOUR.name
FROM BERRY_FLAVOUR
INNER JOIN FLAVOUR ON FLAVOUR.id_flavour =
BERRY_FLAVOUR.id_flavour
GROUP BY BERRY_FLAVOUR.id_flavour, FLAVOUR.name;
```

	count		name
	bigint		character varying (50)
1	30		sweet
2	27		sour
3	30		dry
4	28		bitter
5	28		spicy

7.4 Consulta 4

7.4.1 Solució

```
SELECT ITEM.name_object FROM ITEM
JOIN BUYS ON BUYS.id_object = ITEM.id_object
JOIN CONTAINS_ITEM ON CONTAINS_ITEM.id_object = ITEM.id_object
JOIN SELLS ON SELLS.id_object = ITEM.id_object
GROUP BY ITEM.id_object
ORDER BY COUNT(ITEM.id_object) ASC
LIMIT 1;
```

name_object
character varying (50)
sun stone

7.4.2 Explicació

La consulta selecciona el nom de l'objecte de la taula "ITEM" basat en les seves relacions amb les taules "BUYS", "CONTAINS_ITEM" i "SELLS". S'utilitza un INNER JOIN per combinar les taules basant-se en l'identificador d'objecte. Mitjançant la clàusula GROUP BY, s'agrupen els resultats per identificador d'objecte. L'ordre establert a ORDER BY i la limitació especificada a LIMIT permeten obtenir el nom de l'objecte amb el menor comptador d'identificadors d'objecte associats. Això retorna el nom de l'objecte amb la menor quantitat d'associacions, és a dir, el més escàs.

7.4.3 Validació

La validació d'aquesta consulta és molt simple ja que només cal llistar tots els objectes que tenen els entrenadors, l'inventari de les botigues i el número de vegades que s'ha comprat. Tot seguit, ordenes el llistat en ordre ascendent per veure quin és el item més escàs.

```
SELECT COUNT(ITEM.id_object) as total, ITEM.name_object FROM ITEM
JOIN BUYS ON BUYS.id_object = ITEM.id_object
JOIN CONTAINS_ITEM ON CONTAINS_ITEM.id_object = ITEM.id_object
JOIN SELLS ON SELLS.id_object = ITEM.id_object
GROUP BY ITEM.id_object
ORDER BY TOTAL ASC;
```

total bigint	name_object character varying (50)
45	sun stone
50	coupon 1
65	seal bag
70	tm22
80	tm49
80	full restore

(n'hi ha més, 435 files en total)

7.5 Consulta 5

7.5.1 Solució

```
SELECT SUM(BUYS.discount) as discount FROM BUYS
WHERE BUYS.date_time LIKE '%2020%';
```

discount bigint
12595

S'han donat 12595 de descomptes (en quantitat d'or).

7.5.2 Explicació

La consulta calcula la suma dels descomptes de les compres de la taula "BUYS" realitzades durant l'any 2020. S'utilitza la funció SUM per sumar els valors de la columna "discount". La clàusula WHERE filtra les compres que coincideixen amb el patró de que la data tingui el valor "2020" a la columna "date_time" utilitzant l'operador LIKE. El resultat de la suma es el que es retorna.

7.5.3 Validació

Per validar que la consulta és correcte, he llistat tots els descomptes que conformen la suma a la consulta solució per verificar que el SUM d'aquella consulta m'estigués sumant només descomptes de l'any 2020.

```
SELECT BUYS.discount, BUYS.date_time FROM BUYS
WHERE BUYS.date_time LIKE '%2020%';
```

discount integer	date_time character varying (100)
55	Sun Mar 15 19:47:59 CET 2020
0	Mon Dec 28 16:09:13 CET 2020
0	Mon Mar 16 22:53:00 CET 2020
0	Tue Aug 25 13:50:17 CEST 2020
0	Tue Jun 09 12:56:39 CEST 2020
0	Wed Apr 01 23:32:44 CEST 2020
0	Tue Jul 14 22:42:00 CEST 2020
13	Sat Jan 18 02:55:48 CET 2020

7.6 Consulta 6

7.6.1 Solució 1

```
SELECT * FROM BERRY
ORDER BY BERRY.max_num_harvest DESC
LIMIT 1;
```

id_berry [PK] integer	name character varying (50)	growth_time integer	size_berry integer	smoothness character varying (50)	firmness character varying (50)	max_num_harvest integer	natural_gift_powder integer	soil_dryness integer
156	spelon berry	15	133	35	soft	15	70	8

7.6.2 Solució 2

```
SELECT *
FROM BERRY
WHERE max_num_harvest = (
    SELECT MAX(max_num_harvest)
    FROM BERRY
)
LIMIT 1;
```

id_berry [PK] integer	name character varying (50)	growth_time integer	size_berry integer	smoothness character varying (50)	firmness character varying (50)	max_num_harvest integer	natural_gift_powder integer	soil_dryness integer
156	spelon berry	15	133	35	soft	15	70	8

7.6.3 Explicació

La primera solució, selecciona totes les columnes de la taula BERRY i les ordena en ordre descendent (DESC) segons el valor de la columna max_num_harvest. Finalment, s'estableix una restricció de límit amb "LIMIT 1", que indica que només es retornarà una sola fila com a resultat, és a dir, la fila amb el valor més alt per a la columna max_num_harvest.

La segona solució, selecciona totes les columnes de la taula BERRY on el valor de la columna "max_num_harvest" coincideix amb el valor màxim trobat a través d'una subconsulta. La subconsulta, obté el valor màxim de la columna max_num_harvest de la taula BERRY. A continuació, s'aplica aquest valor màxim com a criteri de selecció a la consulta principal per trobar la fila corresponent. Finalment, s'estableix una restricció de límit amb LIMIT 1 per retornar només una sola fila com a resultat. Això assegura que es retorni la fila amb el valor màxim per a la columna max_num_harvest.

7.6.4 Validació

Puc afirmar que aquesta consulta és correcte ja que si llistes tots els camps de la BERRY i els ordenes en funció del max_num_harvest, veus que te'n surten 5 amb el mateix valor max_num_harvest i el més alt (1). Després, mires el id de cadascun d'aquestes 5 berries per a verificar que quan fas LIMIT 1, la que et surt és una de les que han sortit llistades quan no has limitat la consulta (aquesta és la solució 1). Cal tenir en compte que per defecte, les consultes s'ordenen en funció del id en ordre ascendent.

```
SELECT * FROM BERRY
ORDER BY BERRY.max_num_harvest DESC;
```

Surten 5 berries que tenen el mateix valor de max_num_harvest.

id_berry [PK] integer	name character varying (50)	growth_time integer	size_berry integer	smoothness character varying (50)	firmness character varying (50)	max_num_harvest integer	natural_gift_powder integer	soil_dryness integer
156	spelon berry	15	133	35	soft	15	70	8
157	pamtre berry	15	244	35	very soft	15	70	8
158	watmel berry	15	250	35	soft	15	80	8
159	durin berry	15	280	35	hard	15	80	8
160	belue berry	15	300	35	very soft	15	80	8
152	cornn berry	6	75	30	hard	10	70	10

7.7 Disparador 1

7.7.1 Solució

(Recomanació, aquest disparador ocupa molt espai, millor obrir-lo directament al codi)
No cal crear cap taula addicional per aquest disparador.

```
CREATE OR REPLACE FUNCTION heals()
RETURNS TRIGGER AS $$

BEGIN
    IF EXISTS (SELECT 1
               FROM ITEM AS IT
               JOIN HEAL AS H ON IT.id_object = H.id_heal
               WHERE IT.id_object = old.id_object)
    THEN

        UPDATE POKEMON_CAUGHT AS PC
        SET pokemon_remaining_hp =
            (SELECT ROUND(LEAST (PC.pokemon_remaining_hp +
                                (SELECT H.heal_points
                                 FROM ITEM AS IT
                                 JOIN HEAL AS H ON
                                     IT.id_object = 26)
                                , CASE WHEN
                                    N.id_stat_incremented = BS.id_base_stat AND BS.name = 'hp' THEN ((2*AB.base_value+
                                    (pow(PC.level,2)/4))/100 + 5) * 1.1)
                                    WHEN
                                    N.id_stat_decremented = BS.id_base_stat AND BS.name = 'hp' THEN ((2*AB.base_value+
                                    (pow(PC.level,2)/4))/100 + 5) * 0.9)
                                    ELSE
                                    ((2*AB.base_value+ (pow(PC.level,2)/4))/100 + 5)) END)) prova
            FROM TEAM AS TE
            JOIN TRAINER AS TP ON TE.id_trainer = TP.id_trainer
            JOIN NATURE AS N ON N.id_nature = PC.id_nature
            JOIN POKEMON AS POK ON POK.id_pokemon = PC.id_pokemon
            JOIN ACQUIRE_BASE_STAT AS AB ON AB.id_pokemon =
                POK.id_pokemon
            JOIN BASE_STAT AS BS ON BS.id_base_stat =
                AB.id_base_stat
            JOIN BACKPACK AS BP ON BP.id_trainer = TP.id_trainer
            WHERE TE.slot = 1 and BS.name = 'hp' AND
                AND BP.id_backpack = old.id_backpack)
        WHERE PC.id_pokemon_caught = (SELECT PC2.id_pokemon_caught
                                       FROM POKEMON_CAUGHT AS PC2
                                       JOIN TEAM AS TE ON TE.id_pokemon_caught = PC2.id_pokemon_caught
                                       JOIN TRAINER AS TP ON TE.id_trainer = TP.id_trainer
                                       JOIN BACKPACK AS BP ON BP.id_trainer = TP.id_trainer
                                       WHERE TE.slot = 1 AND BP.id_backpack = old.id_backpack
                                       );

    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS healing ON CONTAINS_ITEM;
CREATE TRIGGER healing
AFTER DELETE ON CONTAINS_ITEM
FOR EACH ROW
EXECUTE FUNCTION heals();
```

7.7.2 Explicació

Aquest disparador el que fa és curar un Pokémon mitjançant qualsevol objecte que estigui considerat a la base de dades com a objecte curatiu. Tot bon entrenador porta a la seva motxilla objectes curatius, i quan el moment ho requereix els utilitzen per curar el seu equip Pokémon. En aquest cas si volen curar a algun Pokémon l'hauran de posar a la primera posició del seu equip Pokémon, per a què aquest

disparador el detecti. Aquest disparador s'executarà cada cop que es faci un "DELETE" a la taula "CONTAINS_ITEM" però està dissenyat per a que si l'objecte el qual ha estat eliminat no és un objecte curatiu, no faci res. Ara bé, si l'objecte és curatiu curarà al Pokémon amb un màxim de la capacitat de curació d'aquell objecte, però si abans succeeix que el Pokémon ja no pot tenir més Punts de Salut, el disparador prioritzarà els punts de salut d'aquell Pokémon.

Pel correcte funcionament d'aquest disparador s'utilitzen taules com ara:

- TRAINER
- POKEMON
- POKEMON_CAUGHT
- TEAM
- NATURE
- BASE_STAT
- ACQUIRE_BASE_STAT
- BACKPACK
- ITEM
- HEAL

A més a més de la taula on es farà la comanda "DELETE" que és "CONTAINS_ITEM".

La lògica d'aquest disparador és bastant complexa, ja que encara que només es faci una "Update" de la taula "POKEMON_CAUGHT" s'ha de tenir moltes altres taules a l'hora de calcular la salut màxima del Pokémon, a més a més que pot variar segons la naturalesa d'aquest sigui, favorable, desfavorable o neutre. D'altra banda les condicions que s'ha de donar perquè aquest càlcul de curació és dui a terme, és només mirar si el "ITEM" es troba dins la taula "HEAL".

7.7.3 Validació

A continuació, unes quantes consultes per poder validar aquest trigger, i també està l'insert que s'ha utilitzat per provar el correcte funcionament d'aquest.

```
SELECT * FROM BASE_STAT;

SELECT * --PC2.id_pokemon_caught
      FROM POKEMON_CAUGHT AS PC2
      JOIN TEAM AS TE ON TE.id_pokemon_caught = PC2.id_pokemon_caught
      JOIN TRAINER AS TP ON TE.id_trainer = TP.id_trainer
      JOIN BACKPACK AS BP ON BP.id_trainer = TP.id_trainer
      WHERE TE.slot = 1 AND
            BP.id_backpack = 260 --old.id_backpack
      ;

-- mira tot de la motxilla
select * from backpack;

select LOWER(PC.pokemon_remaining_hp + H.heal_points
            , CASE WHEN N.id_stat_incremented = BS.id_base_stat AND BS.name = 'hp' THEN
                (((2*AB.base_value+ (pow(PC.level,2)/4))/100 + 5) * 1.1)
                WHEN N.id_stat_decremented = BS.id_base_stat AND BS.name = 'hp' THEN
                (((2*AB.base_value+ (pow(PC.level,2)/4))/100 + 5) * 0.9)
                ELSE (((2*AB.base_value+ (pow(PC.level,2)/4))/100 + 5)) END)
      FROM POKEMON_CAUGHT AS PC
      JOIN TEAM AS TE ON TE.id_pokemon_caught = PC.id_pokemon_caught
      JOIN TRAINER AS TP ON TE.id_trainer = TP.id_trainer
      JOIN NATURE AS N ON N.id_nature = PC.id_nature
      JOIN POKEMON AS POK ON POK.id_pokemon = PC.id_pokemon
      JOIN ACQUIRE_BASE_STAT AS AB ON AB.id_pokemon = POK.id_pokemon
      JOIN BASE_STAT AS BS ON BS.id_base_stat = AB.id_base_stat
      JOIN ITEM AS IT ON IT.id_object =
      JOIN HEAL AS H ON IT.id_object = H.id_heal
      WHERE TE.slot = 1 AND
```

```

        BP.id_backpack = 260 --old.id_backpack
        IT.id_object = 26      --old.id_object
;

```

7.8 Disparador 2

7.8.1 Solució

Cal crear la taula WARNING_TABLE.

```

DROP TABLE IF EXISTS WARNING_TABLE;
CREATE TABLE IF NOT EXISTS WARNING_TABLE (
    id SERIAL PRIMARY KEY,
    id_user VARCHAR(100),
    id_trainer INTEGER,
    table_affected VARCHAR (100),
    warning_message VARCHAR (1000),
    date_time date
);

CREATE OR REPLACE FUNCTION buying_items()
RETURNS TRIGGER AS $$
BEGIN
    IF NOT EXISTS (
        SELECT 1
        FROM TRAINER AS TR
        WHERE new.id_trainer = TR.id_trainer AND TR.pokeCoins_trainer - new.cost >= 0
    ) THEN
        INSERT INTO WARNING_TABLE (id_trainer, warning_message)
        SELECT new.id_trainer,
            format('Trainer #s tried to buy %s but his card has been declined, insufficient funds.',
new.id_trainer, IT.name_object)
        FROM ITEM AS IT
        WHERE new.id_object = IT.id_object
        ;

        RETURN NULL;
    END IF;

    IF NOT EXISTS (
        SELECT 1
        FROM STORE AS ST
        JOIN SELLS AS SL ON SL.id_store = ST.id_store
        WHERE new.id_store = ST.id_store AND new.id_object = SL.id_object AND SL.stock > 0
    ) THEN
        INSERT INTO WARNING_TABLE (id_trainer, warning_message)
        SELECT new.id_trainer,
            format('Trainer #s tried to buy %s at %s store in %s, however the store ran out of stock.',
new.id_trainer, IT.name_object, ST.store_name, AR.area_name)
        FROM ITEM AS IT
        JOIN SELLS AS SL ON SL.id_object = IT.id_object
        JOIN STORE AS ST ON ST.id_store = SL.id_store
        JOIN AREA AS AR ON AR.id_area = ST.id_area
        JOIN CITY AS CY ON CY.id_area = AR.id_area
        WHERE new.id_object = IT.id_object and new.id_store = ST.id_store
        ;

        RETURN NULL;
    END IF;

    UPDATE SELLS SET stock = GREATEST(stock - 1, 0)
    WHERE SELLS.id_store = new.id_store and SELLS.id_object = new.id_object;

    INSERT INTO CONTAINS_ITEM (id_backpack, id_object, obtention_method, date_time)
    VALUES (new.id_trainer, new.id_object, 'BOUGHT', CURRENT_TIMESTAMP);

    UPDATE TRAINER SET pokeCoins_trainer = pokeCoins_trainer - new.cost
    WHERE id_trainer = new.id_trainer;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS buying ON BUYS;

```

```
CREATE TRIGGER buying
AFTER INSERT ON BUYS
FOR EACH ROW
EXECUTE FUNCTION buying_items();
```

7.8.2 Explicació

Aquest disparador el que fa és actualitzar tots els paràmetres després de què un entrenador realitzi una compra a una botiga. Això ho fa a partir d'un nou INSERT a la taula "BUYS" que és una taula, que representa la relació ternària entre "trainer", "store" i "item".

Per al correcte funcionament d'aquest disparador han d'existir les taules:

- WARNING_TABLE
- SELLS
- TRAINER
- ITEM
- STORE
- BACKPACK
- CONTAINS_ITEM
- AREA
- CITY

El que fa la funció és primer de tot si l'entrenador té diners o no, en cas de que no tingui suficients diners, no es podrà realitzar la compra i es crearà un missatge d'error dins de la taula "WARNING_TABLE". Si, sí que té diners per realitzar la compra, s'afegirà a la taula "CONTAINS_ITEM" amb l'identificador de la "BACKPACK" del "TRAINER" el nou "ITEM" comprat sempre que hi hagi disponibilitat d'aquest a la taula "SELLS" relacionada amb "STORE". En cas de que no hi hagi, s'avortarà la compra i se li tornaran els diners a l'entrenador que ha realitzat la compra, en cas contrari la compra s'efectuarà de manera correcte i se li retiraran els diners a l'Entrenador que pertoqui.

7.8.3 Validació

A continuació, unes quantes consultes per poder validar aquest trigger, i també està l'insert que s'ha utilitzat per provar el correcte funcionament d'aquest.

```
SELECT * FROM STORE
WHERE id_store = 31;
```

```
SELECT ITEM.name_object, SELLS.* FROM SELLS
JOIN ITEM ON ITEM.id_object = SELLS.id_object
WHERE id_store = 31 and name_object = 'pp max';
```

```
INSERT INTO BUYS (id_transaction, id_trainer, id_store, id_object, amount, cost, discount,
date_time)
VALUES ((SELECT max(id_transaction) FROM BUYS) + 1, 45, 31, 53, 1, 1800, 0, current_timestamp);
```

```
SELECT * FROM CONTAINS_ITEM
WHERE id_backpack = 45
;
```

```
SELECT 53,
format('Trainer #s tried to buy s at s store in s, however the store ran out of stock.',
45, IT.name_object, ST.store_name, AR.area_name)
FROM ITEM AS IT
JOIN SELLS AS SL ON SL.id_object = IT.id_object
JOIN STORE AS ST ON ST.id_store = SL.id_store
JOIN AREA AS AR ON AR.id_area = ST.id_area
JOIN CITY AS CY ON CY.id_area = AR.id_area
WHERE 53 = IT.id_object and 31 = ST.id_store;
```

```
SELECT 45,
format('Trainer #s tried to buy s but his card has been declined, insufficient funds.',
45, IT.name_object)
FROM ITEM AS IT
```



```
WHERE 45 = IT.id_object;

SELECT 1
FROM STORE AS ST
      JOIN SELLS AS SL ON SL.id_store = ST.id_store
      WHERE 31 = ST.id_store AND 53 = SL.id_object AND SL.stock > 0;

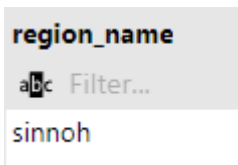
SELECT * FROM WARNING_TABLE;
```

8 Hora d'explorar

8.1 Consulta 1

8.1.1 Solució 1

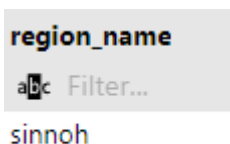
```
SELECT REGION.region_name
FROM REGION
      INNER JOIN AREA ON AREA.id_region = REGION.id_region
      INNER JOIN SUBAREA ON SUBAREA.id_area = AREA.id_area
GROUP BY REGION.region_name
ORDER BY COUNT(SUBAREA.id_subarea) DESC
LIMIT 1;
```



region_name
sinnoh

8.1.2 Solució 2

```
SELECT REGION.region_name
FROM REGION
WHERE REGION.id_region = (
      SELECT AREA.id_region
      FROM AREA
            INNER JOIN SUBAREA ON SUBAREA.id_area = AREA.id_area
      GROUP BY AREA.id_region
      ORDER BY COUNT(SUBAREA.id_subarea) DESC
      LIMIT 1
);
```



region_name
sinnoh

8.1.3 Explicació

La consulta busca trobar el nom de la regió que té més subàrees associades. Per aconseguir-ho, es combinen les taules "REGION", "AREA" i "SUBAREA" utilitzant clàusules d'unió (INNER JOIN) basades en les seves claus primàries i foranes corresponents.

S'utilitza GROUP BY per agrupar els resultats segons el nom de la regió. Després, s'aplica la funció COUNT a la columna "id_subarea" de la taula "SUBAREA" per obtenir el nombre de subàrees per a cada regió.

La clàusula ORDER BY s'emplea per ordenar els resultats en ordre descendent segons el recompte de subàrees. En afegir la clàusula LIMIT 1 al final de la consulta, s'obté únicament la primera fila, que correspon a la regió amb la major quantitat de subàrees.

8.1.4 Validació

La consulta està ben estructurada i lògicament construïda. Utilitza instruccions "JOIN" per combinar adequadament les taules relacionades. La consulta busca la regió amb el major nombre de subàrees associades. Mitjançant l'ús de les clàusules "GROUP BY" i "COUNT", agrupa les regions segons el seu nom i compta el nombre de subàrees per a cada regió. A continuació, ordena els resultats en ordre descendent basant-se en el compte de subàrees i limita el resultat a una única entrada, obtenint així la regió amb el major nombre de subàrees.

```
SELECT REGION.id_region,  
       REGION.region_name,  
       COUNT(SUBAREA.id_subarea) as num_subareas  
FROM REGION  
      INNER JOIN AREA ON AREA.id_region = REGION.id_region  
      INNER JOIN SUBAREA ON SUBAREA.id_area = AREA.id_area  
GROUP BY REGION.id_region,  
         REGION.region_name  
ORDER BY COUNT(SUBAREA.id_subarea) DESC;
```

id_region	region_name	num_subareas
abc Filter...	abc Filter...	abc Filter...
2	sinnoh	156
3	kanto	149
1	hoenn	121
4	johto	101

8.2 Consulta 2

8.2.1 Solució

```
SELECT DISTINCT ON (GYM.gym_name) gym_name,  
       AREA.area_name AS city_name,  
       AREA_ROUTE.area_name AS route_name,  
       A1.area_name AS north,  
       A2.area_name AS east,  
       A3.area_name AS west,  
       A4.area_name AS south  
FROM AREA  
      INNER JOIN CITY ON CITY.id_area = AREA.id_area  
      INNER JOIN GYM ON GYM.id_city = CITY.id_city  
      LEFT JOIN ROUTE ON (  
          ROUTE.north = AREA.id_area  
          OR ROUTE.east = AREA.id_area  
          OR ROUTE.west = AREA.id_area  
          OR ROUTE.south = AREA.id_area  
      )  
      LEFT JOIN AREA AS A1 ON A1.id_area = ROUTE.north  
      LEFT JOIN AREA AS A2 ON A2.id_area = ROUTE.east  
      LEFT JOIN AREA AS A3 ON A3.id_area = ROUTE.west
```

```
LEFT JOIN AREA AS A4 ON A4.id_area = ROUTE.south
LEFT JOIN AREA AS AREA_ROUTE ON AREA_ROUTE.id_area = ROUTE.id_area;
```

gym_name	city_name	route_name	north	east	west	south
Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...
1st Hoenn League Gym	rustboro city	route 115	meteor falls	NULL	NULL	rustboro city
1st Indigo League Gym	pewter city	route 3	NULL	mt moon	pewter city	NULL
1st Johto League Gym	violet city	route 36	route 37	violet city	national park	ruins of alph
1st Sinnoh League Gym	oreburgh city	route 207	route 206	mt coronet	NULL	oreburgh city
2nd Hoenn League Gym	dewford town	route 106	route 105	NULL	NULL	dewford town
2nd Indigo League Gym	cerulean city	route 5	cerulean city	NULL	NULL	saffron city
2nd Johto League Gym	azalea town	route 33	NULL	union cave	azalea town	NULL
2nd Sinnoh League Gym	eterna city	route 205	eterna forest	eterna city	floorama town	NULL
3rd Hoenn League Gym	mauville city	route 117	NULL	mauville city	verdanturf town	NULL
3rd Indigo League Gym	vermillion city	route 6	saffron city	NULL	NULL	vermillion city
3rd Johto League Gym	goldenrod city	route 34	goldenrod city	NULL	NULL	ilex forest
3rd Sinnoh League Gym	veilstone city	route 215	NULL	veilstone city	route 210	NULL
4th Hoenn League Gym	lavaridge town	route 112	NULL	route 111	lavaridge town	NULL
4th Indigo League Gym	celadon city	route 7	NULL	saffron city	celadon city	NULL
4th Johto League Gym	ecruteak city	route 37	ecruteak city	NULL	NULL	route 36
4th Sinnoh League Gym	pastoria city	route 213	valor lakefront	NULL	pastoria city	NULL
5th Hoenn League Gym	petalburg city	route 102	NULL	oldale town	petalburg city	NULL
5th Indigo League Gym	fuchsia city	route 15	NULL	route 14	fuchsia city	NULL
5th Johto League Gym	cianwood city	route 41	NULL	route 40	cianwood city	NULL
5th Sinnoh League Gym	hearthome city	route 208	NULL	hearthome city	mt coronet	NULL
6th Hoenn League Gym	fortree city	route 120	fortree city	NULL	NULL	route 121
6th Indigo League Gym	saffron city	route 7	NULL	saffron city	celadon city	NULL
6th Johto League Gym	olivine city	route 39	route 38	NULL	NULL	olivine city
6th Sinnoh League Gym	canalave city	route 218	NULL	jubilife city	canalave city	NULL
7th Hoenn League Gym	mossdeep city	route 124	NULL	mossdeep city	lilycove city	route 126
7th Indigo League Gym	cinnabar island	route 20	NULL	route 19	cinnabar island	NULL
7th Johto League Gym	mahogany town	route 44	NULL	ice path	mahogany town	NULL
7th Sinnoh League Gym	snowpoint city	NULL	NULL	NULL	NULL	NULL
8th Hoenn League Gym	sootopolis city	NULL	NULL	NULL	NULL	NULL
8th Indigo League Gym	viridian city	route 22	NULL	viridian city	NULL	NULL
8th Johto League Gym	blackthorn city	route 45	blackthorn city	NULL	NULL	route 46
8th Sinnoh League Gym	sunyshore city	route 222	NULL	sunyshore city	valor lakefront	NULL

8.2.2 Explicació

La consulta està dissenyada per obtenir informació detallada sobre els gimnasos i les seves ubicacions geogràfiques. S'utilitzen diverses clàusules d'unió (INNER JOIN i LEFT JOIN) per combinar les taules "AREA", "CITY", "GYM" i "ROUTE" en funció de les seves claus primàries i estrangeres corresponents.

S'utilitza DISTINCT ON per seleccionar registres únics basats en el nom del gimnàs. Això significa que s'obtindrà només un registre per cada nom de gimnàs.

Els LEFT JOIN s'utilitzen per vincular la taula "ROUTE" amb la taula "AREA" en diverses direccions geogràfiques (nord, est, oest, sud), la qual cosa permet obtenir informació addicional sobre les àrees veïnes.

Les subconsultes LEFT JOIN amb les taules "AREA" s'utilitzen per obtenir els noms de les àrees veïnes associades a cada direcció geogràfica (nord, est, oest, sud).

L'objectiu general de la consulta és obtenir un conjunt de resultats que mostri el nom del gimnàs, el nom de la ciutat, el nom de la ruta i els noms de les àrees veïnes en cada direcció geogràfica.

8.2.3 Validació

La consulta està ben estructurada i lògicament construïda. Utilitza instruccions "JOIN" i "LEFT JOIN" per combinar adequadament les taules relacionades. La consulta obté informació detallada sobre els gimnasos i les seves ubicacions, incloent el nom del gimnàs, el nom de la ciutat, el nom de la ruta, les àrees del nord, est, oest i sud associades, i l'àrea de la ruta. Utilitza la clàusula "DISTINCT ON" per seleccionar de manera distintiva el nom del gimnàs. Les unions amb les taules

ROUTE i AREA permeten obtenir les àrees adjacents a cada àrea en cas que hi hagi rutes disponibles.

Mostra bàsicament totes les rutes menys aquelles que la ciutat no esta connectada mitjançant una ruta y aquelles que si, es mostra la ciutat a un dels punts cardinals de la ruta.

8.3 Consulta 3

8.3.1 Solució

```
SELECT id_encounter,
       id_pokemon,
       ENCOUNTERS.id_subarea,
       chance,
       condition_type,
       condition_value,
       method_to_find,
       min_level,
       max_level,
       region_name
FROM ENCOUNTERS
      INNER JOIN SUBAREA ON SUBAREA.id_subarea = ENCOUNTERS.id_subarea
      INNER JOIN AREA ON AREA.id_area = SUBAREA.id_area
      INNER JOIN REGION ON REGION.id_region = AREA.id_region
WHERE ENCOUNTERS.method_to_find LIKE 'surf'
      AND REGION.region_name LIKE 'sinnoh';
```

id_encounter	id_pokemon	id_subarea	chance	condition_type	condition_value	method_to_find	min_level	max_level	region_name
1549	41	382	60			surf	20	30	sinnoh
1550	41	382	30			surf	20	40	sinnoh
1551	41	387	60			surf	20	30	sinnoh
1552	41	387	30			surf	20	40	sinnoh
1556	41	394	60			surf	20	30	sinnoh
1557	41	394	30			surf	20	40	sinnoh
1566	41	425	60			surf	20	30	sinnoh
1576	41	427	60			surf	20	30	sinnoh
1856	42	382	5			surf	30	40	sinnoh
1857	42	382	4			surf	30	40	sinnoh
1858	42	382	1			surf	30	40	sinnoh
1864	42	387	5			surf	30	40	sinnoh
1865	42	387	4			surf	30	40	sinnoh
1866	42	387	1			surf	20	40	sinnoh
1875	42	394	5			surf	30	40	sinnoh
1876	42	394	4			surf	30	40	sinnoh
1877	42	394	1			surf	20	40	sinnoh
1887	42	421	60			surf	35	45	sinnoh
1888	42	421	30			surf	35	45	sinnoh
1889	42	421	5			surf	40	50	sinnoh
1890	42	421	4			surf	40	50	sinnoh
1891	42	421	1			surf	40	50	sinnoh
1895	42	422	60			surf	40	50	sinnoh
1896	42	422	30			surf	45	55	sinnoh
1903	42	425	5			surf	20	40	sinnoh
1904	42	427	5			surf	20	40	sinnoh
2409	54	372	60			surf	20	30	sinnoh
2410	54	372	30			surf	20	40	sinnoh
2421	54	395	4			surf	20	40	sinnoh
2422	54	395	1			surf	20	40	sinnoh
2433	54	396	4			surf	20	40	sinnoh
2434	54	396	1			surf	20	40	sinnoh

8.3.2 Explicació

La consulta està dissenyada per obtenir informació detallada sobre els encontres amb Pokémon en àrees aquàtiques ("surf") de la regió de Sinnoh. S'utilitzen clàusules d'unió (INNER JOIN) per combinar les taules "ENCOUNTERS", "SUBAREA", "AREA" i "REGION" segons les claus primàries i estrangeres corresponents.

Els JOIN s'utilitzen per vincular les taules i obtenir informació relacionada. En aquest cas, s'usen INNER JOIN per assegurar-se que només s'obtinguin registres que tinguin correspondència en totes les taules involucrades.

L'objectiu principal de la consulta és obtenir els encontres amb Pokémon en àrees aquàtiques de la regió de Sinnoh, mostrant detalls com l'ID de l'encontre, l'ID del Pokémon, l'ID de la subàrea, la probabilitat, el tipus de condició, el valor de la condició, el mètode de cerca, els nivells mínim i màxim, i el nom de la regió.

8.3.3 Validació

La consulta està ben estructurada i lògicament construïda. Utilitza instruccions "JOIN" per combinar adequadament les taules relacionades. La consulta selecciona diverses columnes de la taula ENCOUNTERS, incloent l'identificador d'encontre, l'identificador de Pokémon, l'identificador de subàrea, la possibilitat, el tipus de condició, el valor de condició, el mètode de trobar, el nivell mínim i màxim, i el nom de la regió. Utilitza les clàusules "INNER JOIN" per connectar les taules ENCOUNTERS, SUBAREA, AREA i REGION. La clàusula "WHERE" filtra els resultats per trobar els encontres amb el mètode de trobar "surf" i que pertanyen a la regió "Sinnoh".

```
SELECT *  
FROM ENCOUNTERS  
WHERE ENCOUNTERS.method_to_find LIKE 'surf';
```

1984	42	42/	surf	5	20	40
2046	42	260	surf	60	30	35

```
SELECT SUBAREA.id_subarea  
FROM SUBAREA  
    INNER JOIN AREA ON AREA.id_area = SUBAREA.id_area  
    INNER JOIN REGION ON REGION.id_region = AREA.id_region  
WHERE REGION.region_name LIKE 'sinnoh';
```

id_subarea
Filter...
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402

8.4 Consulta 4

8.4.1 Solució

```
SELECT DISTINCT POKEMON.name_pokemon,
                ENCOUNTERS.chance,
                ENCOUNTERS.min_level
FROM ENCOUNTERS
     INNER JOIN POKEMON ON POKEMON.id_pokemon = ENCOUNTERS.id_pokemon
WHERE ENCOUNTERS.chance <= 1
ORDER BY ENCOUNTERS.min_level ASC
LIMIT 1;
```

name_pokemon	chance	min_level
Filter...	Filter...	Filter...
sentret	1	2

8.4.2 Explicació

La consulta està dissenyada per obtenir informació específica sobre els Pokémon trobats en els encontres. S'utilitza una clàusula d'unió (INNER JOIN) per combinar les taules "ENCOUNTERS" i "POKEMON" basant-se en les claus primàries i estrangeres corresponents.

L'objectiu principal de la consulta és obtenir el nom del Pokémon, la probabilitat de trobada i el nivell mínim de trobada per a aquells Pokémon amb una probabilitat de trobada inferior o igual a 1. Els resultats s'ordenen en ordre ascendent segons el nivell mínim de trobada i la consulta es limita a un sol resultat.

8.4.3 Validació

La consulta està ben estructurada i lògicament construïda. Utilitza la instrucció "JOIN" per combinar les taules ENCOUNTERS i POKEMON basant-se en l'identificador de Pokémon. La consulta selecciona el nom del Pokémon, la probabilitat d'encontre i el nivell mínim d'encontre. La clàusula "WHERE" filtra els resultats per mostrar només aquells encontres amb una probabilitat menor o igual a 1. La consulta ordena els resultats pel nivell mínim d'encontre en ordre ascendent i limita els resultats a només un.

8.5 Consulta 5

8.5.1 Solució

```
-- good
UPDATE ENCOUNTERS
SET method_to_find = 'good rod'
WHERE method_to_find = 'good';

-- old
UPDATE ENCOUNTERS
SET method_to_find = 'old rod'
WHERE method_to_find = 'old';
Query returned successfully in 45 msec.

-- super
UPDATE ENCOUNTERS
SET method_to_find = 'super rod'
WHERE method_to_find = 'super';
Query returned successfully in 58 msec.

-- rock
UPDATE ENCOUNTERS
SET method_to_find = (
    SELECT method_to_find
    FROM ENCOUNTERS
    WHERE method_to_find LIKE 'good rod'
    OR method_to_find LIKE 'old rod'
    OR method_to_find LIKE 'super rod'
    ORDER BY RANDOM()
    LIMIT 1
)
WHERE method_to_find = 'rock';
```


Query returned successfully in 30 msec.

8.5.2 Explicació

Les tres primeres sentències UPDATE serveixen per actualitzar els valors de la columna "method_to_find" als valors corresponents ('good rod', 'old rod', 'super rod') segons el seu valor anterior ('good', 'old', 'super').

La quarta sentència UPDATE utilitza una subconsulta per seleccionar de manera aleatòria un dels valors ('good rod', 'old rod', 'super rod') que s'hagin actualitzat prèviament a la columna "method_to_find". Aquesta subconsulta s'ordena de manera aleatòria (ORDER BY RANDOM()) i es limita a un únic resultat (LIMIT 1).

L'objectiu principal de la consulta és actualitzar el valor de la columna "method_to_find" a 'rock', utilitzant de manera aleatòria un dels valors anteriors ('good rod', 'old rod', 'super rod') que s'hagin actualitzat prèviament.

8.5.3 Validació

La consulta està ben estructurada i realitza actualitzacions a la taula ENCOUNTERS de manera adequada. Els primers tres comandaments UPDATE modifiquen el valor de l'atribut "method_to_find" per als encontres que corresponen als mètodes "good", "old" i "super", respectivament. L'últim comandament UPDATE actualitza el valor de l'atribut "method_to_find" per als encontres que tenien el mètode "rock". Aquest últim comandament utilitza una subconsulta per seleccionar aleatòriament un dels mètodes (good rod, old rod o super rod) i assignar-lo com a nou valor per a l'atribut "method_to_find".

```
SELECT COUNT(method_to_find) AS num_good_method
FROM ENCOUNTERS
WHERE method_to_find = 'good';
0
```

```
SELECT COUNT(method_to_find) AS num_good_rod_method
FROM ENCOUNTERS
WHERE method_to_find = 'good rod';
```

```
SELECT COUNT(method_to_find) AS num_old_method
FROM ENCOUNTERS
WHERE method_to_find = 'old';
```

```
SELECT COUNT(method_to_find) AS num_old_rod_method
FROM ENCOUNTERS
WHERE method_to_find = 'old rod';
```

```

SELECT COUNT(method_to_find) AS num_super_method
FROM ENCOUNTERS
WHERE method_to_find = 'super';

```

```

SELECT COUNT(method_to_find) AS num_super_rod_method
FROM ENCOUNTERS
WHERE method_to_find = 'super rod';

```

```

SELECT COUNT(method_to_find) AS num_rock_method
FROM ENCOUNTERS
WHERE method_to_find = 'rock';

```

num_rock_method
0

8.6 Consulta 6

8.6.1 Solució

```

SELECT DISTINCT AREA.area_name
FROM AREA
    INNER JOIN ROUTE ON ROUTE.id_area = AREA.id_area
    INNER JOIN SUBAREA ON SUBAREA.id_area = AREA.id_area
    INNER JOIN ENCOUNTERS ON ENCOUNTERS.id_subarea = SUBAREA.id_subarea
WHERE pavement LIKE 'Grass'
    AND (
        condition_value LIKE '%night%'
        OR condition_value LIKE '%rain%'
    );

```

area_name
abc Filter...
route 1
route 14
route 16
route 201
route 204
route 209
route 212
route 26
route 28
route 34
route 39
route 47
route 5
route 7

8.6.2 Explicació

La consulta té com a objectiu identificar i mostrar els noms de les àrees que compleixen dues condicions específiques: ser classificades com a zones de gespa ("Grass") i tenir trobades amb condicions relacionades amb la nit o la pluja. S'utilitzen clàusules de junta per unir les taules rellevants i assegurar la correspondència de claus.

L'ús de la clàusula DISTINCT garanteix que només s'obtinguin noms d'àrees únics, evitant repeticions en els resultats.

8.6.3 Validació

La consulta està ben estructurada i construïda de manera lògica. Utilitza l'instrucció "JOIN" per combinar les taules AREA, ROUTE, SUBAREA i ENCOUNTERS de manera adequada, basant-se en les claus corresponents. La clàusula "WHERE" filtra els resultats per mostrar només els noms de les àrees que tenen un paviment "Grass" i compleixen amb la condició de tenir en el seu valor de condició les paraules "night" o "rain". En utilitzar l'operador lògic "OR", se seleccionen les àrees que compleixen almenys una de les dues condicions especificades.

```
SELECT AREA.id_area, SUBAREA.id_subarea
FROM AREA
    INNER JOIN ROUTE ON ROUTE.id_area = AREA.id_area
    INNER JOIN SUBAREA ON SUBAREA.id_area = AREA.id_area
WHERE paviment LIKE 'Grass';
```

```
SELECT *
FROM ENCOUNTERS
WHERE condition_value LIKE '%night%'
    OR condition_value LIKE '%rain%';
```

id_encounter	id_pokemon	id_subarea	method_to_find	condition_type	condition_value	chance	min_level	max_level
297	13	28	walk	time	night	10	5	5
348	14	28	walk	time	night	10	3	3
358	14	28	walk	time	night	1	7	7
398	15	28	walk	time	night	4	10	10
548	19	182	walk	time	night	10	2	2
551	19	182	walk	time	night	5	4	4
552	19	183	walk	time	night	30	3	3
553	19	183	walk	time	night	10	4	4
554	19	184	walk	time	night	30	4	4
555	19	184	walk	time	night	10	5	5
562	19	211	walk	time	night	10	3	3
569	19	211	walk	time	night	4	5	5
570	19	211	walk	time	night	1	5	5
579	19	212	walk	time	night	10	3	3
586	19	212	walk	time	night	4	5	5
587	19	212	walk	time	night	1	5	5
590	19	185	walk	time	night	30	4	4
603	19	186	walk	time	night	30	7	7
604	19	186	walk	time	night	20	6	6
606	19	186	walk	time	night	10	6	6
616	19	151	walk	time	night	30	14	14
618	19	151	walk	time	night	10	16	16
625	19	213	walk	time	night	10	22	22
628	19	213	walk	time	night	5	23	23
633	19	213	walk	time	night	4	24	24
634	19	213	walk	time	night	1	24	24
643	19	214	walk	time	night	10	22	22
646	19	214	walk	time	night	5	23	23
651	19	214	walk	time	night	4	24	24
652	19	214	walk	time	night	1	24	24
661	19	215	walk	time	night	10	22	22
664	19	215	walk	time	night	5	23	23

8.7 Disparador 1

8.7.1 Solució

```

CREATE
OR REPLACE FUNCTION create_city_trigger() RETURNS TRIGGER AS $$
DECLARE
    lasalia_id INT;
    leader_id INT;
BEGIN
    INSERT INTO region (region_name) VALUES ('Lasalià')
    ON CONFLICT (region_name) DO NOTHING;

    SELECT AREA.id_area INTO lasalia_id
    FROM AREA
    JOIN REGION ON AREA.id_region = REGION.id_region
    WHERE region_name = 'Lasalià';

    UPDATE CITY SET id_area = lasalia_id
    WHERE id_city = new.id_city;

    INSERT INTO trainer (id_trainer, name_trainer, class_trainer,
pokeCoins_trainer, exp_trainer)
    VALUES ((SELECT max(id_trainer) + 1 FROM TRAINER), 'Laureano', 'Líder de
Gimnasio', 1932420,3248523);

    SELECT id_trainer INTO leader_id FROM trainer WHERE name_trainer = 'Laureano';

```

```

        INSERT INTO gym (id_gym, id_city, id_type, id_trainer, gym_name,
gym_badge_leader)
        VALUES ((SELECT max(id_city) + 1 FROM CITY),new.id_city, 4, leader_id, 'Trigger
Gym', 'Trigger Badge' );

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS city_trigger ON CITY;
CREATE TRIGGER city_trigger
AFTER
INSERT ON city FOR EACH ROW
EXECUTE
    FUNCTION create_city_trigger();

```

8.7.2 Explicació

Aquest codi defineix una funció de disparador en PostgreSQL anomenada "create_city_trigger" que s'activa després de cada inserció a la taula "city". Quan es fa una inserció a la taula "city", la funció s'executa automàticament. Dins de la funció, es realitzen diverses operacions en seqüència.

Primer, s'insereix una nova regió anomenada "Lasalià" a la taula "region", sempre que no hi hagi conflicte amb un nom de regió existent.

A continuació, es consulta l'ID de l'àrea corresponent a la regió "Lasalià" a través de les taules "area" i "region". Aquest ID s'utilitza per actualitzar l'àrea de la ciutat recentment inserida a la taula "city".

Seguidament, s'insereix un nou entrenador a la taula "trainer" amb un ID generat automàticament, un nom, una classe, una quantitat de monedes i una experiència específiques. Aquest entrenador es defineix com a líder del gimnàs.

Després, es consulta l'ID de l'entrenador "Laureano" a través de la taula "trainer". Aquest ID s'utilitza per inserir un nou gimnàs a la taula "gym" amb l'ID de la ciutat recentment inserida, un ID de tipus determinat, l'ID de l'entrenador, un nom de gimnàs i un distintiu de gimnàs.

Finalment, es retorna el nou registre inserit a la taula "city". En resum, aquest disparador crea automàticament una nova regió, assigna l'àrea correcta a les ciutats inserides i afegeix un gimnàs amb un líder designat cada vegada que s'insereix una nova ciutat.

8.7.3 Validació

```
INSERT INTO CITY (id_city, population)
VALUES ((SELECT max(id_city) + 1 FROM CITY), 283946)
;
```

```
SELECT * FROM AREA
JOIN REGION ON AREA.id_region = REGION.id_region
WHERE REGION.region_name = 'Lasalià';
```

```
SELECT * FROM CITY
ORDER BY id_city DESC;
```

```
SELECT GYM.* FROM GYM
JOIN CITY ON CITY.id_city = GYM.id_city
JOIN AREA ON AREA.id_area = CITY.id_area
JOIN REGION ON AREA.id_region = REGION.id_region
WHERE REGION.region_name = 'Lasalià';
```

8.8 Disparador 2

8.8.1 Solució

```
CREATE
OR REPLACE FUNCTION check_pokemon_caught_trigger() RETURNS TRIGGER AS $$
    DECLARE
        pokemon_row RECORD;
        encounter_row RECORD;
        route_row RECORD;
        trainer_row RECORD;
    BEGIN
        SELECT *
        INTO pokemon_row
        FROM POKEMON
        WHERE id_pokemon = NEW.id_pokemon;

        SELECT *
        INTO encounter_row
        FROM ENCOUNTERS
        WHERE id_pokemon = NEW.id_pokemon;

        SELECT ROUTE.*, AR.area_name
        INTO route_row
        FROM ROUTE
        JOIN AREA AS AR ON AR.id_area = ROUTE.id_area
        JOIN SUBAREA AS SU ON SU.id_area = AR.id_area
        WHERE id_subarea = NEW.id_subarea;
```

```

SELECT *
INTO trainer_row
FROM TRAINER
WHERE id_trainer = NEW.id_trainer;

IF lower(route_row.pavement) <> 'water' AND
(lower(NEW.obtaining_method) <> 'walk' OR lower(NEW.obtaining_method) <>
'headbutt') THEN
    INSERT INTO WARNING_TABLE (id_trainer, warning_message)
    VALUES (trainer_row.id_trainer, 'Trainer ' ||
trainer_row.name_trainer || ' captured ' || pokemon_row.name_pokemon || ' in ' ||
route_row.area_name || ', with either incorrect method, level, or capture chance
1.');
```

```

    ELSIF lower(route_row.pavement) = 'Water' AND
lower(NEW.obtaining_method) <> 'surf' THEN
    INSERT INTO WARNING_TABLE (id_trainer, warning_message)
    VALUES (trainer_row.id, 'Trainer ' || trainer_row.name_trainer || '
captured ' || pokemon_row.name_pokemon || ' in ' || route_row.area_name || ', with
either incorrect method, level, or capture chance 2.');
```

```

    ELSIF encounter_row.chance <= 0 OR (NEW.level NOT BETWEEN
encounter_row.min_level AND encounter_row.max_level) THEN
    INSERT INTO WARNING_TABLE (id_trainer, warning_message)
    VALUES (trainer_row.id_trainer, 'Trainer ' ||
trainer_row.name_trainer || ' captured ' || pokemon_row.name_pokemon || ' in ' ||
route_row.area_name || ', with either incorrect method, level, or capture chance
3.');
```

```

    END IF;

RETURN NEW;
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS check_pokemon_caught ON POKEMON_CAUGHT;
CREATE TRIGGER
    check_pokemon_caught BEFORE
INSERT
    ON POKEMON_CAUGHT FOR EACH ROW
EXECUTE
    FUNCTION check_pokemon_caught_trigger();

```

8.8.2 Explicació

Aquest codi defineix una funció de disparador en PostgreSQL anomenada "check_pokemon_caught_trigger". Aquesta funció s'executa abans de cada inserció a la taula "pokemon_caught".

Quan es realitza una inserció a la taula "pokemon_caught", la funció es crida automàticament. Dins de la funció, s'efectuen diverses operacions. Es realitzen consultes a diverses taules, com "pokemon", "encounters", "route" i "trainer", per obtenir informació sobre el Pokémon capturat, el trobament, la ruta i l'entrenador associat.

A continuació, es realitzen diverses comprovacions condicionals. Es verifica si el paviment de la ruta és diferent de "water" i si el mètode d'obtenció del Pokémon no és "walk" o "headbutt", o bé, si el paviment de la ruta és "water" i el mètode d'obtenció no és "surf". En aquests casos, s'afegeix una entrada a la taula "warning_table" amb un missatge d'advertència indicant que el jugador ha capturat incorrectament el Pokémon.

També es verifica si la probabilitat de trobada és menor o igual a zero o si el nivell del Pokémon capturat està fora de l'interval de nivells de trobada. En aquests casos, s'afegeix una entrada a la taula "warning_table" amb un missatge d'advertència que indica que el jugador ha capturat el Pokémon amb mètode, nivell o probabilitat incorrectes.

Finalment, es retorna el nou registre inserit a la taula "pokemon_caught". En resum, aquest disparador s'encarrega de validar les dades de captura dels Pokémon i generar missatges d'advertència en cas de detectar dades incorrectes.

8.8.3 Validació

```
SELECT * FROM POKEMON_CAUGHT;
```

```
SELECT ROUTE.*, AR.area_name, EN.*
FROM ROUTE
JOIN AREA AS AR ON AR.id_area = ROUTE.id_area
JOIN SUBAREA AS SU ON SU.id_area = AR.id_area
JOIN ENCOUNTERS AS EN ON EN.id_subarea = SU.id_subarea
WHERE SU.id_subarea = 479 and
id_pokemon = 418;
-- ERROR --
INSERT INTO POKEMON_CAUGHT (id_pokemon_caught, id_trainer, nick_name, xp, level,
gender, id_subarea, date_caught, pokeball_used_caught, obtaining_method,
pokemon_status_condition, pokemon_max_hp, pokemon_remaining_hp, id_pokemon,
id_nature, id_pokemon_ability, id_object, id_move1, id_move2, id_move3, id_move4)
VALUES ((SELECT max(id_pokemon_caught) + 1 FROM
POKEMON_CAUGHT),0,'Shorty',4681,15,'female',479,current_timestamp,'great
ball','surf', 'freeze',8,0,418,19,33,510,152,NULL,NULL,NULL);

-- CORRECTO --
```



```
INSERT INTO POKEMON_CAUGHT (id_pokemon_caught, id_trainer, nick_name, xp, level,  
gender, id_subarea, date_caught, pokeball_used_caught, obtaining_method,  
pokemon_status_condition, pokemon_max_hp, pokemon_remaining_hp, id_pokemon,  
id_nature, id_pokemon_ability, id_object, id_move1, id_move2, id_move3, id_move4)  
VALUES ((SELECT max(id_pokemon_caught) + 1 FROM  
POKEMON_CAUGHT),50,'Greta',4681,44,'female',496,current_timestamp,'great  
ball','surf', 'freeze',8,0,319,19,33,510,152,NULL,NULL,NULL);
```

```
SELECT * FROM WARNING_TABLE;  
DELETE FROM WARNING_TABLE;
```

```
SELECT * FROM POKEMON_CAUGHT WHERE id_pokemon_caught >= 13716;  
DELETE FROM POKEMON_CAUGHT WHERE id_pokemon_caught >= 13716;
```

9 Preguntes creuades

9.1 Consulta 1

9.1.1 Solució

```
SELECT REGION.region_name
FROM REGION
    INNER JOIN ORGANIZATION ON ORGANIZATION.id_region = REGION.id_region
    INNER JOIN EVIL_TRAINER ON EVIL_TRAINER.id_organization =
ORGANIZATION.id_organization
    INNER JOIN TRAINER AS evil ON evil.id_trainer = EVIL_TRAINER.id_trainer
    INNER JOIN BATTLE ON BATTLE.trainer_winner = evil.id_trainer
    INNER JOIN TRAINER AS good ON good.id_trainer = BATTLE.trainer_loser
GROUP BY REGION.region_name
ORDER BY COUNT(good.id_trainer) DESC
LIMIT 1;

noenn
```

9.1.2 Explicació

Aquesta consulta està dissenyada per obtenir el nom de la regió amb més victòries dels entrenadors bons contra els entrenadors dolents. Es fan servir juntes internes (INNER JOIN) per combinar les taules "REGION", "ORGANIZATION", "EVIL_TRAINER", "TRAINER", "BATTLE" i "good" basant-se en les claus primàries i estrangeres corresponents.

L'ús de la clàusula GROUP BY permet agrupar els resultats pel nom de la regió. Això permet calcular el nombre de victòries dels entrenadors bons a cada regió.

L'ús de l'ORDER BY COUNT(good.id_trainer) DESC permet ordenar els resultats de manera descendent segons el nombre de victòries dels entrenadors bons.

Finalment, la clàusula LIMIT 1 assegura que s'obté només el primer registre, que correspon a la regió amb el major nombre de victòries dels entrenadors bons.

9.1.3 Validació

La consulta està ben estructurada i construïda de manera lògica. Utilitza l'instrucció "JOIN" per combinar les taules REGION, ORGANIZATION, EVIL_TRAINER, TRAINER i BATTLE de manera adequada, basant-se en les claus corresponents. La clàusula "WHERE" filtra els resultats per obtenir la regió on els entrenadors malvats han guanyat més batalles. S'agrupen els resultats pel nom de la regió i s'ordenen de manera descendent segons la quantitat d'entrenadors bons derrotats. Finalment, el resultat es limita a mostrar només la regió amb la major quantitat de derrotes pels entrenadors bons.

```
SELECT REGION.region_name,
    COUNT(good.id_trainer)
FROM REGION
    INNER JOIN ORGANIZATION ON ORGANIZATION.id_region = REGION.id_region
    INNER JOIN EVIL_TRAINER ON EVIL_TRAINER.id_organization =
ORGANIZATION.id_organization
    INNER JOIN TRAINER AS evil ON evil.id_trainer = EVIL_TRAINER.id_trainer
```

```

    INNER JOIN BATTLE ON BATTLE.trainer_winner = evil.id_trainer
    INNER JOIN TRAINER AS good ON good.id_trainer = BATTLE.trainer_loser
GROUP BY REGION.region_name
ORDER BY COUNT(good.id_trainer) DESC;

```

region_name	count
abc Filter...	abc Filter...
hoenn	52
johto	30
kanto	20
sinnoh	11

9.2 Consulta 2

9.2.1 Solució

```

(
    SELECT DISTINCT POKEMON.name_pokemon
    FROM POKEMON
        INNER JOIN POKEMON_CAUGHT ON POKEMON_CAUGHT.id_pokemon = POKEMON.id_pokemon
)
EXCEPT (
    SELECT DISTINCT POKEMON.name_pokemon
    FROM POKEMON
        INNER JOIN ENCOUNTERS ON ENCOUNTERS.id_pokemon = POKEMON.id_pokemon
);

```

name_pokemon

Filter...

combusken
 bellossom
 poliwrath
 luxray
 wartortle
 ampharos
 clefable
 giratina-altered
 huntail
 dusknoir
 vileplume
 hitmontop
 skorupi
 scizor
 Swalot
 aerodactyl
 tyranitar
 drapion
 flareon
 glalie
 feebas
 jolteon
 lopunny
 staraptor
 prinplup
 venusaur
 cradily
 togekiss
 sceptile
 shedinja
 mismagius
 slowking
 blastoise
 Shelgon
 pidgeot
 anorith
 porygon2
 kabutops

9.2.2 Explicació

Aquesta consulta està dissenyada per obtenir els noms dels Pokémon que han estat capturats (registrats a la taula POKEMON_CAUGHT) però que no han estat trobats en cap trobada (registrats a la taula ENCOUNTERS). Es fa servir l'operador EXCEPT per comparar els resultats de dues subconsultes.

La primera subconsulta obté els noms dels Pokémon que han estat capturats, fent servir una clàusula INNER JOIN per combinar les taules "POKEMON" i "POKEMON_CAUGHT" basant-se en les claus primàries i foranes corresponents.

La segona subconsulta obté els noms dels Pokémon que han estat trobats en alguna trobada, fent servir una clàusula INNER JOIN per combinar les taules "POKEMON" i "ENCOUNTERS" basant-se en les claus primàries i foranes corresponents.

L'operador EXCEPT permet obtenir els noms dels Pokémon que apareixen a la primera subconsulta però no a la segona, és a dir, els Pokémon capturats però no trobats. Això permet identificar quins Pokémon encara no s'han trobat en cap trobada.

9.2.3 Validació

La consulta utilitza la clàusula EXCEPT per obtenir la llista de Pokémon que han estat capturats però no han estat trobats en cap trobada. La subconsulta dins de l'EXCEPT selecciona els Pokémon capturats, mentre que la subconsulta després de l'EXCEPT selecciona els Pokémon trobats en les trobades. En utilitzar l'EXCEPT, es realitza una diferència entre les dues llistes, retornant només els Pokémon capturats que no han estat trobats en trobades.

9.3 Consulta 3

9.3.1 Solució

```
DELETE FROM ENCOUNTERS USING POKEMON,  
        EVOLVES  
WHERE POKEMON.id_pokemon = EVOLVES.id_pokemon_base  
      AND ENCOUNTERS.id_pokemon = POKEMON.id_pokemon  
      AND ENCOUNTERS.method_to_find LIKE 'only'  
RETURNING *;
```

id_encounter	id_pokemon	id_subarea	method_to_find	condition_type	condition_value	chance	min_level	max_level	id_pokemon (1)
5543	100	62	only			100	40	40	100

9.3.2 Explicació

Aquesta consulta està dissenyada per eliminar registres de la taula "ENCOUNTERS" que compleixin certes condicions, utilitzant les taules "POKEMON" i "EVOLVES" per trobar les dades relacionades.

Es fa servir la clàusula DELETE FROM per eliminar registres de la taula "ENCOUNTERS".

Es fa servir la clàusula USING per especificar les taules "POKEMON" i "EVOLVES" i establir les condicions de les clàusules WHERE.

Es fa servir la clàusula WHERE per establir les condicions de coincidència entre les taules "POKEMON", "EVOLVES" i "ENCOUNTERS".

Es fa servir la condició ENCOUNTERS.method_to_find LIKE 'only' per filtrar els registres de la taula "ENCOUNTERS" que tenen el valor 'only' a la columna "method_to_find".

Es fa servir la clàusula RETURNING * per obtenir els registres eliminats com a resultat de la consulta.

9.3.3 Validació

La consulta està ben estructurada i utilitza la clàusula DELETE FROM per eliminar registres de la taula ENCOUNTERS. Utilitza les taules POKEMON i EVOLVES per realitzar la condició d'eliminació, assegurant-se que l'id_pokemon de POKEMON coincideixi amb l'id_pokemon_base de EVOLVES. A més, es filtra per l'atribut method_to_find a ENCOUNTERS per seleccionar només aquells registres que continguin la paraula "only". La clàusula RETURNING * s'utilitza per retornar els registres eliminats com a resultat de la consulta.

```
SELECT POKEMON.name_pokemon  
FROM ENCOUNTERS  
      INNER JOIN POKEMON ON POKEMON.id_pokemon = ENCOUNTERS.id_pokemon  
      INNER JOIN EVOLVES ON EVOLVES.id_pokemon_base = POKEMON.id_pokemon  
WHERE ENCOUNTERS.method_to_find LIKE 'only';
```







name_pokemon
abc Filter...

No data

9.4 Consulta 4

9.4.1 Solució

```
SELECT TRAINER.name_trainer,  
       POKEMON.name_pokemon,  
       M1.name_move AS move_1,  
       M2.name_move AS move_2,  
       M3.name_move AS move_3,  
       M4.name_move AS move_4  
FROM POKEMON  
     INNER JOIN POKEMON_CAUGHT ON POKEMON_CAUGHT.id_pokemon = POKEMON.id_pokemon  
     INNER JOIN TRAINER ON TRAINER.id_trainer = POKEMON_CAUGHT.id_trainer  
     INNER JOIN GYM ON GYM.id_trainer = TRAINER.id_trainer  
     INNER JOIN TYPE_POKEMON AS T1 ON T1.id_type <> GYM.id_type  
AND POKEMON.id_type_1 = T1.id_type  
     INNER JOIN TYPE_POKEMON AS T2 ON T2.id_type <> GYM.id_type  
AND POKEMON.id_type_2 = T2.id_type  
     INNER JOIN POKEMON_MOVE AS M1 ON M1.id_move = POKEMON_CAUGHT.id_move1  
     INNER JOIN POKEMON_MOVE AS M2 ON M2.id_move = POKEMON_CAUGHT.id_move2  
     INNER JOIN POKEMON_MOVE AS M3 ON M3.id_move = POKEMON_CAUGHT.id_move3  
     INNER JOIN POKEMON_MOVE AS M4 ON M4.id_move = POKEMON_CAUGHT.id_move4  
WHERE TRAINER.class_trainer LIKE '%Gym Leader%';
```

name_trainer	name_pokemon	move_1	move_2	move_3	move_4
 Filter...	 Filter...	 Filter...	 Filter...	 Filter...	 Filter...
Misty	tailow	flail	sharpen	substitute	conversion 2
Brock	spearow	smokescreen	take down	false swipe	peck
Brock	bronzor	confusion	light screen	amnesia	metal claw
Brock	mawile	metal claw	iron tail	sweet kiss	charm
Brock	latias	kinesis	mirror coat	light screen	barrier
Brock	pelipper	waterfall	sky attack	hydro pump	crabhammer
Brock	charizard	ember	peck	mirror move	fly
Misty	bronzor	confusion	teleport	barrier	future sight
Misty	froslass	night shade	curse	haze	mist
Misty	swablu	wing attack	present	thrash	mega punch
Misty	gloom	poison gas	stun spore	vine whip	poison powder
Erika	forretress	steel wing	megahorn	spider web	leech life
Erika	nincada	magnitude	bone club	mud slap	fury cutter
Erika	gallade	light screen	dream eater	cross chop	vital throw
Erika	aggron	metal claw	ancient power	rock slide	rock throw
Erika	swampert	bubble beam	fissure	bone rush	withdraw
Erika	gible	dragon rage	earthquake	magnitude	mud slap
Koga	meditite	mirror coat	confusion	psywave	double kick
Koga	spiritomb	feint attack	thief	confuse ray	beat up
Koga	corsola	clamp	ancient power	octazooka	bubble beam
Koga	omastar	octazooka	waterfall	clamp	rollout
Koga	spearow	swagger	body slam	skull bash	sweet scent
Sabrina	roserade	poison gas	smog	solar beam	sleep powder
Sabrina	hoothoot	fury swipes	slam	mega kick	attract
Sabrina	tyranitar	rock throw	ancient power	beat up	rock slide
Sabrina	piloswine	bonemerang	icy wind	magnitude	spikes
Sabrina	omanyte	sandstorm	clamp	ancient power	bubble
Sabrina	tailow	growl	struggle	spike cannon	tackle
Sabrina	nidoking	toxic	poison gas	poison powder	fissure
Sabrina	roserade	poison sting	cotton spore	spore	stun spore
Sabrina	noctowl	sketch	hidden power	supersonic	encore
Sabrina	mamoswine	spikes	ice beam	ice punch	magnitude
Sabrina	empoleon	withdraw	steel wing	bubble	metal claw
Sabrina	combee	wing attack	fly	fury cutter	twineedle
Sabrina	vespiquen	leech life	string shot	twineedle	aeroblast
Sabrina	mantyke	hydro pump	octazooka	wing attack	whirlpool
Blaine	slowking	hypnosis	agility	rest	mirror coat
Blaine	dragonite	twister	gust	aeroblast	peck

9.4.2 Explicació

Aquesta consulta està dissenyada per obtenir informació sobre els entrenadors que són líders de gimnasos i els seus Pokémon capturats, juntament amb els moviments associats a cada Pokémon.

Es fan servir clàusules d'unió (INNER JOIN) per combinar les taules "POKEMON", "POKEMON_CAUGHT", "TRAINER", "GYM", "TYPE_POKEMON" i "POKEMON_MOVE" en base a les claus primàries i foranes corresponents.

L'objectiu principal de la consulta és obtenir els noms dels entrenadors (TRAINER.name_trainer), els noms dels Pokémon capturats (POKEMON.name_pokemon) i els noms dels moviments associats a cada Pokémon (M1.name_move, M2.name_move, M3.name_move, M4.name_move).

S'utilitzen múltiples clàusules d'unió per vincular les taules i establir les condicions de coincidència. No es fa servir una clàusula GROUP BY, ja que no hi ha una agrupació específica en aquesta consulta.

9.4.3 Validació

La consulta està ben estructurada i utilitza diverses clàusules JOIN per combinar múltiples taules de manera adequada. La clàusula WHERE filtra els resultats per mostrar només aquells entrenadors que tenen la classe "Gym Leader". Es seleccionen els noms de l'entrenador, el nom del Pokémon capturat i els noms dels quatre moviments associats al Pokémon capturat. A més, s'estableixen condicions addicionals a les clàusules JOIN per garantir que els tipus de Pokémon siguin diferents al tipus del gimnàs.

9.5 Consulta 5

9.5.1 Solució

```
SELECT TRAINER.name_trainer,
       POKEMON.name_pokemon
FROM TRAINER
      INNER JOIN EVIL_TRAINER ON EVIL_TRAINER.id_trainer = TRAINER.id_trainer
      INNER JOIN ORGANIZATION ON ORGANIZATION.id_organization =
EVIL_TRAINER.id_organization
      INNER JOIN POKEMON_CAUGHT ON POKEMON_CAUGHT.id_trainer = TRAINER.id_trainer
      INNER JOIN POKEMON ON POKEMON.id_pokemon = POKEMON_CAUGHT.id_pokemon
      INNER JOIN ENCOUNTERS ON ENCOUNTERS.id_pokemon = POKEMON.id_pokemon
      INNER JOIN POKEMON_MOVE AS M1 ON M1.id_move = POKEMON_CAUGHT.id_move1
      INNER JOIN POKEMON_MOVE AS M2 ON M2.id_move = POKEMON_CAUGHT.id_move2
      INNER JOIN POKEMON_MOVE AS M3 ON M3.id_move = POKEMON_CAUGHT.id_move3
      INNER JOIN POKEMON_MOVE AS M4 ON M4.id_move = POKEMON_CAUGHT.id_move4
WHERE TRAINER.class_trainer LIKE '%Leader%'
      AND ENCOUNTERS.method_to_find LIKE 'gift'
      AND (
          M1.effect LIKE '%Drain%'
          OR M2.effect LIKE '%Drain%'
          OR M3.effect LIKE '%Drain%'
          OR M4.effect LIKE '%Drain%'
      );
```

9.5.2 Explicació

Aquesta consulta està dissenyada per obtenir els noms dels entrenadors (TRAINER.name_trainer) i els noms dels Pokémon (POKEMON.name_pokemon) que compleixen diverses condicions específiques. Es fan servir clàusules d'unió (INNER JOIN) per combinar les taules "TRAINER", "EVIL_TRAINER", "ORGANIZATION", "POKEMON_CAUGHT", "POKEMON", "ENCOUNTERS" i "POKEMON_MOVE" en base a les claus primàries i foranes corresponents.

L'objectiu principal de la consulta és obtenir els noms dels entrenadors i els noms dels Pokémon que compleixen les següents condicions: l'entrenador és un líder (TRAINER.class_trainer LIKE '%Leader%'), el mètode de trobar els Pokémon és "gift" (ENCOUNTERS.method_to_find LIKE 'gift'), i almenys un dels moviments associats al Pokémon té l'efecte "Drain" (M1.effect LIKE '%Drain%', M2.effect LIKE '%Drain%', M3.effect LIKE '%Drain%', M4.effect LIKE '%Drain%').

9.5.3 Validació

La consulta està ben estructurada i utilitza clàusules JOIN per combinar diverses taules de manera adequada. La clàusula WHERE filtra els resultats per mostrar només aquells entrenadors que tenen la paraula "Leader" en la seva classe, els encontres que tenen el mètode "gift" i els moviments que contenen la paraula "Drain" en el seu efecte. Es seleccionen els noms de l'entrenador i del Pokémon capturat, juntament amb els moviments associats a cada Pokémon.

```
SELECT ENCOUNTERS.method_to_find,
       M1.effect AS move1,
       M2.effect AS move2,
       M3.effect AS move3,
       M4.effect AS move4
FROM ENCOUNTERS
     INNER JOIN POKEMON ON POKEMON.id_pokemon = ENCOUNTERS.id_pokemon
     INNER JOIN POKEMON_CAUGHT ON POKEMON_CAUGHT.id_pokemon = POKEMON.id_pokemon
     INNER JOIN POKEMON_MOVE AS M1 ON M1.id_move = POKEMON_CAUGHT.id_move1
     INNER JOIN POKEMON_MOVE AS M2 ON M2.id_move = POKEMON_CAUGHT.id_move2
     INNER JOIN POKEMON_MOVE AS M3 ON M3.id_move = POKEMON_CAUGHT.id_move3
     INNER JOIN POKEMON_MOVE AS M4 ON M4.id_move = POKEMON_CAUGHT.id_move4
WHERE ENCOUNTERS.method_to_find LIKE 'gift'
     AND (
         M1.effect LIKE '%Drain%'
         OR M2.effect LIKE '%Drain%'
         OR M3.effect LIKE '%Drain%'
         OR M4.effect LIKE '%Drain%'
     );
```


method_to_find	move1	move2	move3	move4
Filter...	Filter...	Filter...	Filter...	Filter...
gift	Drains half the damage inflicted to heal the user.	Seeds the target, stealing HP from it every turn.	Puts the target to sleep.	Puts the target to sleep.
gift	Drains half the damage inflicted to heal the user.	Seeds the target, stealing HP from it every turn.	Puts the target to sleep.	Puts the target to sleep.
gift	Has an increased chance for a critical hit.	Drains half the damage inflicted to heal the user.	Lowers the target's Speed by two stages.	Inflicts regular damage with no additional
gift	Has an increased chance for a critical hit.	Drains half the damage inflicted to heal the user.	Lowers the target's Speed by two stages.	inflicts regular damage with no additional
gift	Drains half the damage inflicted to heal the user.	Paralyzes the target.	Has an increased chance for a critical hit.	Lowers the target's Speed by two stages.
gift	Drains half the damage inflicted to heal the user.	Paralyzes the target.	Has an increased chance for a critical hit.	Lowers the target's Speed by two stages.
gift	Drains half the damage inflicted to heal the user.	Inflicts regular damage with no additional effect.	Seeds the target, stealing HP from it every turn.	Drains half the damage inflicted to heal th
gift	Drains half the damage inflicted to heal the user.	Inflicts regular damage with no additional effect.	Seeds the target, stealing HP from it every turn.	Drains half the damage inflicted to heal th
gift	Puts the target to sleep.	Requires a turn to charge before attacking.	Drains half the damage inflicted to heal the user.	Lowers the target's Speed by two stages.
gift	Puts the target to sleep.	Requires a turn to charge before attacking.	Drains half the damage inflicted to heal the user.	Lowers the target's Speed by two stages.
gift	Drains half the damage inflicted to heal the user.	Heals the user by half its max HP. Affected by weather.	Drains half the damage inflicted to heal the user.	Seeds the target, stealing HP from it every
gift	Drains half the damage inflicted to heal the user.	Heals the user by half its max HP. Affected by weather.	Drains half the damage inflicted to heal the user.	Seeds the target, stealing HP from it every
gift	Heals the user by half its max HP. Affected by weather.	Drains half the damage inflicted to heal the user.	Has a Seffect, chance's chance to poison the target.	Hits every turn for 2-3 turns, then confuse
gift	Heals the user by half its max HP. Affected by weather.	Drains half the damage inflicted to heal the user.	Has a Seffect, chance's chance to poison the target.	Hits every turn for 2-3 turns, then confuse
gift	Has a Seffect, chance's chance to lower the target's Special ...	Seeds the target, stealing HP from it every turn.	Drains half the damage inflicted to heal the user.	Badly poisons the target, inflicting more d
gift	Inflicts regular damage with no additional effect.	Puts the target to sleep.	Drains half the damage inflicted to heal the user.	Heals the user by half its max HP. Affected
gift	Inflicts regular damage with no additional effect.	Puts the target to sleep.	Drains half the damage inflicted to heal the user.	Heals the user by half its max HP. Affected
gift	Confuses the target.	Lowers the target's Attack by two stages.	Heals the user by half its max HP. Affected by weath...	Drains half the damage inflicted to heal th
gift	Heals the user by half its max HP. Affected by weather.	Drains half the damage inflicted to heal the user.	Puts the target to sleep.	Paralyzes the target.
gift	Heals the user by half its max HP. Affected by weather.	Drains half the damage inflicted to heal the user.	Puts the target to sleep.	Paralyzes the target.
gift	Drains half the damage inflicted to heal the user.	Paralyzes the target.	Hits every turn for 2-3 turns, then confuses the user.	Lowers the target's Speed by two stages.
gift	Drains half the damage inflicted to heal the user.	Paralyzes the target.	Hits every turn for 2-3 turns, then confuses the user.	Lowers the target's Speed by two stages.
gift	Poisons the target.	Drains half the damage inflicted to heal the user.	Lowers the target's Speed by two stages.	Seeds the target, stealing HP from it every
gift	Poisons the target.	Drains half the damage inflicted to heal the user.	Lowers the target's Speed by two stages.	Seeds the target, stealing HP from it every
gift	Requires a turn to charge before attacking.	Paralyzes the target.	Drains half the damage inflicted to heal the user.	Hits every turn for 2-3 turns, then confuse
gift	Requires a turn to charge before attacking.	Paralyzes the target.	Drains half the damage inflicted to heal the user.	Hits every turn for 2-3 turns, then confuse
gift	Drains half the damage inflicted to heal the user.	Inflicts regular damage with no additional effect.	Puts the target to sleep.	Has an increased chance for a critical hit.
gift	Drains half the damage inflicted to heal the user.	Inflicts regular damage with no additional effect.	Puts the target to sleep.	Has an increased chance for a critical hit.
gift	Puts the target to sleep.	Hits every turn for 2-3 turns, then confuses the user.	Requires a turn to charge before attacking.	Drains half the damage inflicted to heal th
gift	Puts the target to sleep.	Hits every turn for 2-3 turns, then confuses the user.	Requires a turn to charge before attacking.	Drains half the damage inflicted to heal th
gift	Has a Seffect, chance's chance to raise all of the user's stats ...	Drains half the damage inflicted to heal the user.	Hits twice in the same turn. Has a Seffect, chance's ...	Lowers the target's Speed by two stages.
gift	Has a Seffect, chance's chance to raise all of the user's stats ...	Drains half the damage inflicted to heal the user.	Hits twice in the same turn. Has a Seffect, chance's ...	Lowers the target's Speed by two stages.
gift	Paralyzes the target.	Heals the user by half its max HP. Affected by weather.	Puts the target to sleep.	Drains half the damage inflicted to heal th
gift	Paralyzes the target.	Heals the user by half its max HP. Affected by weather.	Puts the target to sleep.	Drains half the damage inflicted to heal th
gift	Paralyzes the target.	Puts the target to sleep.	Drains half the damage inflicted to heal the user.	Requires a turn to charge before attacking
gift	Paralyzes the target.	Puts the target to sleep.	Drains half the damage inflicted to heal the user.	Requires a turn to charge before attacking

9.6 Consulta 6

9.6.1 Solució

```

SELECT ITEM.name_object,
       TRAINER.name_trainer,
       POKEMON.name_pokemon
FROM POKEMON
     INNER JOIN EVOLVES ON (
         EVOLVES.id_pokemon_base = POKEMON.id_pokemon
         OR EVOLVES.id_pokemon_final = POKEMON.id_pokemon
     )
     INNER JOIN POKEMON_CAUGHT ON POKEMON_CAUGHT.id_pokemon = POKEMON.id_pokemon
     INNER JOIN TRAINER ON TRAINER.id_trainer = POKEMON_CAUGHT.id_trainer
     INNER JOIN BUYS ON BUYS.id_trainer = TRAINER.id_trainer
     INNER JOIN ITEM ON (
         ITEM.id_object = BUYS.id_object
         AND ITEM.id_object = EVOLVES.id_item
     )
WHERE ITEM.name_object = (
    SELECT ITEM.name_object
    FROM POKEMON
         INNER JOIN EVOLVES ON EVOLVES.id_pokemon_base = POKEMON.id_pokemon
         INNER JOIN POKEMON_CAUGHT ON POKEMON_CAUGHT.id_pokemon =
POKEMON.id_pokemon
         INNER JOIN TRAINER ON TRAINER.id_trainer = POKEMON_CAUGHT.id_trainer
         INNER JOIN BUYS ON BUYS.id_trainer = TRAINER.id_trainer
         INNER JOIN ITEM ON ITEM.id_object = BUYS.id_object
         AND ITEM.id_object = EVOLVES.id_item
    GROUP BY ITEM.name_object
    ORDER BY COUNT(TRAINER.id_trainer) DESC
    LIMIT 1
)
GROUP BY ITEM.name_object,
       TRAINER.name_trainer,

```

```
POKEMON.name_pokemon
HAVING COUNT(TRAINER.id_trainer) > 0;
```

name_object	name_trainer	name_pokemon
abc Filter...	abc Filter...	abc Filter...
moon stone	Beverley	delcatty
moon stone	Eladia	nidoking
moon stone	Emerson	skitty
moon stone	Jerrell	wigglytuff
moon stone	Jonah	clefairy
moon stone	Jonah	delcatty
moon stone	Neal	clefable
moon stone	Pedro	jigglypuff
moon stone	Rueben	nidorino
moon stone	Rupert	clefairy
moon stone	Shad	jigglypuff
moon stone	Tamie	wigglytuff
moon stone	Taylor	skitty
moon stone	William	clefable

9.6.2 Explicació

Aquesta consulta està dissenyada per obtenir els noms dels objectes (ITEM.name_object), dels entrenadors (TRAINER.name_trainer) i dels Pokémon (POKEMON.name_pokemon) relacionats amb una condició específica. Es fan servir clàusules d'unió (INNER JOIN) per combinar les taules "POKEMON", "EVOLVES", "POKEMON_CAUGHT", "TRAINER", "BUYS" i "ITEM" en base a les claus primàries i foranes corresponents.

L'objectiu principal de la consulta és obtenir els noms dels objectes, dels entrenadors i dels Pokémon que compleixen les següents condicions: l'objecte està relacionat amb una evolució de Pokémon (EVOLVES.id_item = ITEM.id_object), l'objecte està comprat per algun entrenador (BUYS.id_object = ITEM.id_object), i l'objecte té el mateix nom que l'objecte més comprat segons una subconsulta.

La clàusula GROUP BY s'utilitza per agrupar els resultats pel nom de l'objecte, el nom de l'entrenador i el nom del Pokémon. Això permet mostrar les dades agrupades de manera lògica.

S'utilitza una subconsulta per obtenir el nom de l'objecte més comprat. Aquesta subconsulta té una clàusula GROUP BY i ORDER BY per calcular el nombre de vegades que cada objecte ha estat comprat i seleccionar el més popular (amb el nombre més gran de compres).

9.6.3 Validació

La consulta està ben estructurada i utilitza clàusules JOIN per combinar diverses taules de manera adequada. Es seleccionen els noms dels objectes, els entrenadors i els Pokémon. La condició JOIN a la taula ITEM assegura que l'objecte estigui relacionat tant amb la compra feta per l'entrenador com amb l'evolució del Pokémon. La clàusula WHERE filtra els resultats per mostrar només aquells registres en què el nom de l'objecte coincideixi amb el nom de l'objecte seleccionat a la subconsulta.

A més, s'aplica un HAVING per mostrar només aquells registres que tinguin un recompte d'entrenadors major a zero.

```
SELECT ITEM.name_object,
       COUNT(TRAINER.id_trainer)
FROM POKEMON
     INNER JOIN EVOLVES ON EVOLVES.id_pokemon_base = POKEMON.id_pokemon
     INNER JOIN POKEMON_CAUGHT ON POKEMON_CAUGHT.id_pokemon = POKEMON.id_pokemon
     INNER JOIN TRAINER ON TRAINER.id_trainer = POKEMON_CAUGHT.id_trainer
     INNER JOIN BUYS ON BUYS.id_trainer = TRAINER.id_trainer
     INNER JOIN ITEM ON ITEM.id_object = BUYS.id_object
     AND ITEM.id_object = EVOLVES.id_item
GROUP BY ITEM.name_object
ORDER BY COUNT(TRAINER.id_trainer) DESC;
```

name_object	count
abc Filter...	abc Filter...
moon stone	7
leaf stone	6
fire stone	5
dubious disc	5
kings rock	5
thunder stone	4
shiny stone	4
deep sea scale	3
deep sea tooth	3
dusk stone	3
electirizer	2
oval stone	2
dawn stone	2
reaper cloth	1
metal coat	1
razor claw	1

9.7 Consulta 7

9.7.1 Solució

```
SELECT POKEMON_CAUGHT.id_pokemon_caught,
       POKEMON.weight_pokemon,
       POKEMON.height_pokemon,
       (((2 * ACQUIRE_BASE_STAT.base_value + (GROWTH_RATE.level_pokemon ^ 2) / 4) /
100) + 5) * (CASE WHEN NATURE.id_stat_incremented = BASE_STAT.id_base_stat THEN
1.15
              WHEN NATURE.id_stat_decremented = BASE_STAT.id_base_stat THEN 0.85
              ELSE 1
              END) AS stat
```

```

FROM POKEMON_CAUGHT
    INNER JOIN POKEMON ON POKEMON.id_pokemon = POKEMON_CAUGHT.id_pokemon
    INNER JOIN NATURE ON NATURE.id_nature = POKEMON_CAUGHT.id_nature
    INNER JOIN GROWTH_RATE ON GROWTH_RATE.id_growth_rate = POKEMON.id_growth_rate
    INNER JOIN ACQUIRE_BASE_STAT ON ACQUIRE_BASE_STAT.id_pokemon =
POKEMON.id_pokemon
    INNER JOIN BASE_STAT ON BASE_STAT.id_base_stat = ACQUIRE_BASE_STAT.id_base_stat
WHERE date_part('year', POKEMON_CAUGHT.date_caught) BETWEEN 2022 AND 2023
GROUP BY POKEMON_CAUGHT.id_pokemon_caught,
    POKEMON.weight_pokemon,
    POKEMON.height_pokemon,
    stat;

```

id_pokemon_cau...	weight_pokemon	height_pokemon	stat
Filter...	Filter...	Filter...	Filter...
3332	1150	10	16.84
6132	2200	25	14.3225
6953	1550	19	6.96
8370	1105	11	16.61
1766	400	14	20.16
8967	1000	20	8.834875
9787	590	18	14.14
13440	600	20	12.96
9787	590	18	11.51
7588	3	3	18.4025
6608	720	6	21.099125
10012	1200	9	11.7225
4857	2050	19	12.75
8967	1000	20	13.73
714	3360	42	20.7225
6953	1550	19	12.96
11824	650	10	11.96
6132	2200	25	8.45
11514	402	15	8.5025
10348	1050	6	9.94000000000001
9889	208	8	11.525875
9148	3360	42	24.44
2661	120	8	10.89
11593	240	8	29.7225
1292	2160	52	30.16
12252	5500	16	9.2625
4588	6830	54	17.009999999999998
11883	2200	21	19.558500000000002
10120	1305	20	29.5625
13440	600	20	7.6
10148	152	6	12.412125
440	270	14	30.5225

9.7.2 Explicació

Aquesta consulta està dissenyada per obtenir les dades dels Pokémon capturats (POKEMON_CAUGHT) juntament amb el seu pes (POKEMON.weight_pokemon), alçada (POKEMON.height_pokemon) i una fórmula per calcular la seva estadística (stat). Es fan servir clàusules d'unió (INNER JOIN) per combinar diverses taules, com ara "POKEMON", "NATURE", "GROWTH_RATE", "ACQUIRE_BASE_STAT" i "BASE_STAT", utilitzant les claus primàries i foranes corresponents.

La clàusula GROUP BY s'utilitza per agrupar els resultats segons l'identificador del Pokémon capturat (POKEMON_CAUGHT.id_pokemon_caught), el seu pes i alçada

(POKEMON.weight_pokemon i POKEMON.height_pokemon), així com l'estadística calculada (stat). Això permet mostrar les dades agrupades de manera lògica i evitar duplicats.

9.7.3 Validació

La consulta està ben estructurada i utilitza clàusules JOIN per combinar diverses taules de manera adequada. Es seleccionen diversos camps relacionats amb el Pokémon capturat, incloent el seu ID, pes i alçada. A més, es calcula una estadística modificada utilitzant fórmules i condicions específiques. S'utilitzen les taules NATURE, GROWTH_RATE, ACQUIRE_BASE_STAT i BASE_STAT per realitzar els càlculs necessaris. La clàusula WHERE filtra els resultats per mostrar només els registres en els quals la data de captura del Pokémon es trobi entre els anys 2022 i 2023. Finalment, s'utilitza la clàusula GROUP BY per agrupar els resultats per l'ID del Pokémon capturat, el seu pes, alçada i l'estadística calculada.

```
SELECT *
FROM POKEMON_CAUGHT
WHERE date_part('year', POKEMON_CAUGHT.date_caught) BETWEEN 2022 AND 2023;
```

id_pokemon_cau...	id_trainer	nick_name	xp	level	id_subarea	date_caught	pokeball_used_e...	obtaining_method	pokemon_status...
Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...	Filter...
24	1	Flakey	3892	17	483	2022-11-29	luxury ball	walk	NULL
27	1	Benson	48	2	395	2022-03-24	ultra ball	walk	NULL
32	1	Flopsy	228	5	190	2023-02-13	safari ball	walk	NULL
68	2	Hunter	269	6	298	2022-05-14	heavy ball	gift	NULL
73	2	Elwood	2489	12	482	2022-12-13	timer ball	walk	NULL
94	3	Samantha	5261	16	495	2022-03-07	nest ball	walk	NULL
96	3	Tango	2378	12	372	2022-12-15	great ball	good rod	NULL
118	3	Butterscotch	4667	17	187	2022-04-28	dusk ball	gift egg	NULL
115	3	Buffy	1050	8	371	2022-07-25	lure ball	surf	NULL
151	5	Gidget	188	5	395	2023-02-10	luxury ball	walk	NULL
160	6	Piedrasantas	2829	11	495	2022-04-28	repeat ball	walk	none
201	7	Pirate	4399	16	484	2022-05-28	lure ball	walk	NULL
214	8	Pippin	9684	18	NULL	2023-03-24	dive ball	super rod spots	NULL
261	10	Bongo	31	1	361	2023-04-07	moon ball	only one	NULL
292	11	Waddles	390	7	508	2022-01-05	dusk ball	walk	NULL
293	12	Digger	45	2	NULL	2022-02-05	friend ball	walk	none
305	12	Winston	5907	17	383	2023-03-04	dusk ball	walk	NULL
334	14	Fulgencio	5265	16	497	2022-12-10	cherish ball	walk	none
352	14	Hannah	1387	9	NULL	2022-11-30	nest ball	gift	NULL
405	16	Jess	3384	15	421	2022-02-19	quick ball	walk	NULL
408	17	Tinkler	2113	14	478	2022-11-10	love ball	walk	none
440	18	Old Glory	1528	9	165	2022-08-06	luxury ball	walk	NULL
452	18	Duchess	642	7	174	2022-11-09	moon ball	walk	NULL
491	19	Yaka	11137	19	182	2022-01-05	great ball	headbutt high	NULL
507	19	Commando	4269	16	383	2022-09-20	ultra ball	walk	NULL
509	19	Bam-bam	253	4	513	2022-08-13	luxury ball	super rod	NULL
512	20	Bo	5443	18	NULL	2023-03-24	heal ball	gift	none
518	20	Queen	30	2	497	2022-12-16	nest ball	walk	NULL
523	20	Freeway	6633	16	NULL	2022-12-18	quick ball	gift	NULL
569	21	Ming	2754	15	12	2022-06-24	poke ball	gift	NULL
572	21	Hamlet	1597	10	51	2022-02-19	master ball	walk	NULL
580	22	Purdy	2055	11	NULL	2023-02-25	poke ball	gift	NULL

9.8 Consulta 8

9.8.1 Solució

```
SELECT DISTINCT ON (TRAINER.id_trainer) TRAINER.id_trainer,
    TRAINER.name_trainer
FROM TRAINER
    INNER JOIN BUYS ON BUYS.id_trainer = TRAINER.id_trainer
    INNER JOIN STORE ON STORE.id_store = BUYS.id_store
    INNER JOIN SELLS ON SELLS.id_store = STORE.id_store
    INNER JOIN ITEM ON (
        ITEM.id_object = SELLS.id_object
        AND ITEM.id_object = BUYS.id_object
    )
    INNER JOIN AREA ON AREA.id_area = STORE.id_area
    INNER JOIN POKEMON_CAUGHT ON POKEMON_CAUGHT.id_trainer = TRAINER.id_trainer
    INNER JOIN POKEMON ON POKEMON.id_pokemon = POKEMON_CAUGHT.id_pokemon
```

```
INNER JOIN EVOLVES ON (
    EVOLVES.id_pokemon_base = POKEMON.id_pokemon
    OR EVOLVES.id_pokemon_final = POKEMON.id_pokemon
);
```

id_trainer	name_trainer
1	Genie
2	Myrna
3	Tod
4	Quintin
5	Elfreda
6	Earnestine
8	Casey
9	Ying
10	Neomi
13	Buster
18	Mark
19	Rueben
21	Rossie
24	Wally
25	Tawanna
31	Sheba
38	Antonio
40	Jonathan
41	Gabrielle
42	Tosha
48	Orlando
49	Justin
50	Carlos
51	Jeffery
53	Lola
55	Kacey
59	Josiah
60	Rayna
66	Shelley
69	Reuben
71	Irma
73	Ivory

9.8.2 Explicació

Aquesta consulta està dissenyada per obtenir una llista distinta d'entrenadors (TRAINER) juntament amb les seves dades d'identificador (TRAINER.id_trainer) i nom (TRAINER.name_trainer). Es fa servir la clàusula DISTINCT ON per retornar només una fila per cada identificador d'entrenador.

Es fan servir clàusules d'unió (INNER JOIN) per combinar diverses taules, com ara "TRAINER", "BUYS", "STORE", "SELLS", "ITEM", "AREA", "POKEMON_CAUGHT", "POKEMON" i "EVOLVES", utilitzant les claus primàries i foranes corresponents.

9.8.3 Validació

La consulta està ben estructurada i utilitza diverses clàusules JOIN per combinar diferents taules de manera adequada. Es seleccionen els camps ID i nom de l'entrenador. La clàusula DISTINCT ON s'utilitza per mostrar només una fila per a cada ID d'entrenador. Es realitzen les connexions necessàries amb les taules BUYS, STORE, SELLS, ITEM, AREA, POKEMON_CAUGHT, POKEMON i EVOLVES per obtenir la informació requerida.

10 Conclusions

10.1 Recursos emprats

Etap	Joan	Andrea	Pol	Leo	Total
Actualització dels models Entitat-Relació i Relacional	0h	0h	0h	0h	0h
Selecció del tipus de dades	0h	0h	0h	0h	0h
Codificació del model relacional	0h	0h	0h	0h	0h
Importació de la base de dades	15min	0h	0h	0h	15min
Implementació i validació de les consultes i disparadors	14h	12h	14h	20h	60h
Memòria	2h	2.5h	2.5h	2.5h	9.5h
Total:	16.25h	14.5h	16.5h	22h	69.75h

Més o menys tots hem necessitat el mateix nombre d'hores per realitzar la seva part. Cal dir però que hi ha hores que algun dels companys pot haver dedicat a ajudar a algú altre i per tant, no ha estat treballant en el seu mòdul.

Les petites desviacions en termes d'hores que es reflecteixen a la taula, venen donades per la diferència de nivell de les consultes de cada mòdul.

10.2 Us d'IA (si cal, 1-2 pàgines)

No aplica. Tots els dubtes que s'han tingut s'han preguntat a classe tant al professorat com als becaris.

10.3 Lliçons apreses i conclusions (1 pàgina)

En aquesta tercera fase hem pogut refrescar els coneixements que vam aprendre en el seu dia a classe sobre com fer consultes a una base de dades i com crear disparadors. Tots els membres de l'equip estem d'acord que realitzar consultes en un projecte de grans dimensions com aquest, no ha sigut una tasca senzilla però tot i així, ho hem aconseguit i ens trobem molt satisfets amb la feina realitzada en aquesta última fase.

Duent a terme aquesta tercera fase, no només hem refrescat i consolidat els coneixements que vam aprendre el primer semestre de l'assignatura sinó que també hem après noves maneres de treballar i de fer consultes que desconexíem o que no havíem utilitzat mai.

Relacionat amb la tercera fase del projecte, volem afegir que aquesta ens ha servit també per comprovar la bona feina que vam fer des de l'inici. Si no haguéssim treballat correctament en les fases anteriors, se'ns hagués fet mol difícil, per no dir impossible, obtenir els resultats correctes de cadascuna de les consultes proposades.

Un cop finalitzat el projecte, ens parem a pensar i veiem tot allò que hem après durant aquest curs a les classes de bases de dades. Quan tots vam començar, gairebé ni sabíem què era una bbdd i molt menys com funcionava i ara, uns mesos després, no ens podem considerar experts però sí que tenim la sensació de que hem assolit una gran varietat de coneixements que n'estem segurs que farem servir al llarg del nostre camí universitari.

Ens agradaria afegir que tots coincidim en que la realització d'un projecte com el PokeSallianWorld ha sigut molt interessant ja que ens ha permès, un cop vistos una varietat de conceptes i

funcionalitats de les bases de dades, poder demostrar tot allò que hem après i aplicar els mencionats coneixements en una base de dades creada per nosaltres. Treballar d'aquesta manera ens ha semblat més entretingut ja que sembla més realista que no pas les activitats d'avaluació contínua que vam realitzar al primer semestre.

Així doncs, considerem que la bona feina realitzada durant aquests mesos, ha obtingut els seus fruits i és per això que entreguem aquest treball molt orgullosos de nosaltres com a equip de treball.