

Projecte de base de dades

PokéSallian World 2022-2023 – Fase 2

Llistat de membres (nom i correu):

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

Andrea Ballester Grifol: andrea.ballester@students.salle.url.edu

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

Leonardo Ruben Edenak Chouev: leonardoruben.edenak@students.salle.url.edu

Data d'entrega: 30 de abril de 2023

Taula de continguts

1	INTRODUCCIÓ (1 PÀGINA).....	3
2	ACTUALITZACIÓ DEL MODEL ENTITAT-RELACIÓ.....	4
2.1	MÒDUL POKÉMONS.....	4
2.2	MÒDUL ENTRENADOR	5
2.3	MÒDUL OBJECTES	7
2.4	MÒDUL EXPLORACIÓ.....	8
3	ACTUALITZACIÓ DEL MODEL RELACIONAL.....	12
3.1	MÒDUL POKÉMONS.....	12
3.2	MÒDUL ENTRENADOR	12
3.3	MÒDUL OBJECTES	15
3.4	MÒDUL EXPLORACIÓ.....	16
4	SELECCIÓ DEL TIPUS DE DADES (1-2 PÀGINES).....	18
5	CODIFICACIÓ DEL MODEL FÍSIC	25
6	IMPORTACIÓ DE LA BASE DE DADES (10-12 PÀGINES).....	25
6.1	ABILITIES.CSV	26
6.2	BATTLE_STATISTICS.CSV	26
6.3	BATTLES.CSV	27
6.4	BERRIES.CSV.....	29
6.5	BERRIES_FLAVOURS.CSV.....	29
6.6	DAMAGE_RELATIONS.CSV.....	29
6.7	ENCOUNTERS.CSV	30
6.8	EVOLUTIONS.CSV	31
6.9	GROWTH_RATES.CSV	33
6.10	GYMS.CSV	33
6.11	ITEMS.CSV	35
6.12	LOCATIONS.CSV	35

6.13	MOVES.CSV	38
6.14	NATURES.CSV	40
6.15	POKEMON.CSV	41
6.16	POKEMON_ABILITIES.CSV	42
6.17	POKEMON_INSTANCES.CSV	43
6.18	POKETEAMS.CSV	44
6.19	PURCHASES.CSV	45
6.20	ROUTES.CSV	45
6.21	STATS.CSV	46
6.22	STOREITEMS.CSV	47
6.23	STORES.CSV	48
6.24	TRAINERITEMS.CSV	48
6.25	TRAINERS.CSV	48
6.26	TYPES.CSV	49
6.27	VILLAINOUS_ORGANIZATIONS.CSV	50
7	VALIDACIÓ DE LA BASE DE DADES	51
7.1	POKÉMONS (1-2 PÀGINES)	51
7.2	ENTRENADORS (1-2 PÀGINES)	55
7.3	COMPRES (1-2 PÀGINES)	64
7.4	EXPLORACIÓ (1-2 PÀGINES)	73
8	CONCLUSIONS	87
8.1	RECURSOS EMPRATS	87
8.2	US D'IA (SI CAL, 1-2 PÀGINES)	87
8.3	LLIÇONS APRESES I CONCLUSIONS (1 PÀGINA)	87

1 Introducció (1 pàgina)

En aquesta fase del projecte se'ns demana implementar una base de dades local en llenguatge SQL en el Sistema de Gestió de Bases de Dades Relacionals (RDBMS) PostgreSQL per emmagatzemar informació dels fitxers “.csv” que ens han proporcionat.

Per començar s'haurà d'instal·lar el programa PgAdmin 4, que és un IDE que permet crear una base de dades en local utilitzant PostgreSQL que és exactament el que es vol.

En el cas de Linux, s'haurà de descarregar el repositori de PostgreSQL i utilitzar la terminal per fer ús d'aquest.

Un cop ja es té les eines per poder desenvolupar la base de dades, es començarà creant una base de dades on crearem les taules d'aquesta, de tal forma que quedin idèntiques als models relacional i conceptual de la fase anterior del projecte. En cas de trobar alguna incongruència o error a l'hora de muntar els models anteriors, al passar-ho a model físic s'han de corregir, tot marcant els canvis que es fan per poder recordar tots el que es va posar en primera instància. Posteriorment, s'ha de fer els canvis necessaris (afegir, modificar o eliminar) les taules perquè coincideixin amb els csv proporcionats i que les dades desades siguin correctes i tinguin sentit per més tard poder fer ús d'aquestes amb facilitat.

Com a desenvolupadors de software i futurs enginyers informàtics, ens han posat alguns requisits per poder crear la nostra base de dades de manera autònoma. Per tant, un dels requisits és inserir les dades de forma manual a les taules que haurem de crear. És a dir, no es podrà utilitzar programes externs com el TOAD o eines que no sigui un script SQL propi. Per poder-ho dur a terme les idees que s'han plantejat com a grup és fer ús de taules temporals on llegir els fitxers “.csv” i després parcejar les dades i inserir-les a les taules finals del disseny.

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

2.1 Mòdul Pokémons

Els diferents canvis que s'han hagut de fer respecte la primera entrega, han quedat reflectits en la segona entrega de la fase 1. És per això que es recomana mirar la memòria de la reentrega de la fase 1 per observar els canvis i explicacions.

2.2 Mòdul Entrenador

En el mòdul entrenador s'han modificat les següents entitats (respecte la entrega sobre 10! A la entrega sobre 8 aquests canvis ja venen inclosos):

- **Battle:** S'ha canviat de relació a entitat
 - o **id_battle:** S'ha afegit una PK, ja que el fitxer “.csv” ens demana que es guardi un identificador per cada combat.
 - o **experience_battle:** S'ha afegit també aquest atribut per poder guardar-nos l'experiència acumulada en batalla de cada entrenador en aquell combat.
- **Team:**
 - o **id_team:** S'ha afegit aquesta PK, pel fet que aquesta taula no complia amb la norma d'atomicitat que ha de complir qualsevol taula d'una base de dades relacional. Tenint abans com a PK la composició dels 6 pokemon i l'entrenador que formaven aquesta taula.
 - o **team_slot:** S'ha afegit per poder diferenciar a quin slot pertany cada pokemon.
 - o **team_number:** S'ha afegit per poder diferenciar el nombre de l'equip de la PK d'aquesta taula.
 - o **Pokemon(1..6)_team:** S'ha esborrat a causa del fet que generava problemes amb aquesta taula perquè tota la taula era un conjunt de (PK/FK) i no complia la norma d'atomicitat d'una taula de base de dades relacional.
- **Trainer:**
 - o **item_gift_leader:** És un camp que s'omplirà en cas que sigui l'entrenador sigui líder de gimnàs i que s'ha de guardar del “.csv”
- **Organization:**
 - o **building_organization:** S'ha afegit, ja que es necessitava per poder guardar l'edifici de l'organització malèvola.
 - o **Headquarter:** S'ha afegit per poder guardar la localització de l'edifici de l'organització malèvola.

En el mòdul entrenador s'han modificat les següents relacions:

- **Trainer has leader Gym (1:1):** Aquesta relació s'ha creat per suplantar la taula Gym Leader, una taula que en un principi havia estat creada com una generalització d'un entrenador que tenia com a atribut extra l'“item_gift_leader” que dona un líder de gimnàs. Al final s'ha acabat decidint que la classe Trainer adoptarà aquest atribut i serà NULL en cas que no sigui líder de gimnàs.
- **Battle has battle_statistics with PokemonCaught (1:N):** Aquesta relació s'ha creat principalment per poder implementar el fitxer “battle_statistics.csv” un fitxer amb les estadístiques de batalla de cada Pokémon que havia participat en ella, així com els punts de vida restants d'aquell Pokémon i el mal rebut i produït per ell. Aquesta relació en ajuntar una batalla amb certa quantitat de Pokémon serà una relació 1 a N. A més a més aquesta relació també guarda els següents atributs.
 - o **damage_recived:** quantitat de mal rebut en aquella batalla per un Pokémon en concret
 - o **damage_dealt:** quantitat de mal produït en aquella batalla per un Pokémon en concret

2.3 Mòdul Objectes

Els diferents canvis que s'han hagut de fer respecte la primera entrega, han quedat reflectits en la segona entrega de la fase 1. És per això que es recomana mirar la memòria de la reentrega de la fase 1 per observar els canvis i explicacions.

2.4 Mòdul Exploració

En el mòdul exploració s'han modificat les següents entitats (respecte la entrega sobre 10! A la entrega sobre 8 aquests canvis ja venen inclosos):

- **Pokemon Caught:**
 - o gender: S'ha afegit aquest atribut per guardar el camp gènere dels Pokémon que apareix en el fitxer “.csv”
- **Route:**
 - o **pavement:** S'ha afegit aquest atribut a causa del fet que apareix en el CSV “routes” i és un camp necessari.
- **Encounters:** S'ha creat aquesta taula que abans era una relació N:M amb les entitats Subarea i Pokemon, però realment no era una relació N:M si dues 1:N amb aquesta taula per entremig.
 - o **chance:** S'ha afegit aquest atribut que ens dona la probabilitat en percentatge que aparegui aquell Pokémon. S'ha afegit a causa del fet que apareix en el CSV “encounters”.
 - o **condition_type:** Atribut que prèviament existia a la relació, però s'ha canviat de nom. Aquesta representa el tipus de condició perquè aparegui el Pokémon.
 - o **condition_value:** Atribut que prèviament existia a la relació, però s'ha canviat de nom. Aquesta representa el valor de condició perquè aparegui el Pokémon.
 - o **method_to_find:** Atribut que prèviament existia a la relació, però s'ha canviat de nom. Aquesta representa el tipus de mètode perquè aparegui el Pokémon.
 - o **min_level:** S'ha afegit aquest atribut que ens dona el nivell mínim amb el qual pot aparèixer el Pokémon. S'ha afegit a causa del fet que apareix en el CSV “encounters”.
 - o **max_level:** S'ha afegit aquest atribut que ens dona el nivell màxim amb el qual pot aparèixer el Pokémon. S'ha afegit pel fet que apareix en el CSV “encounters”.

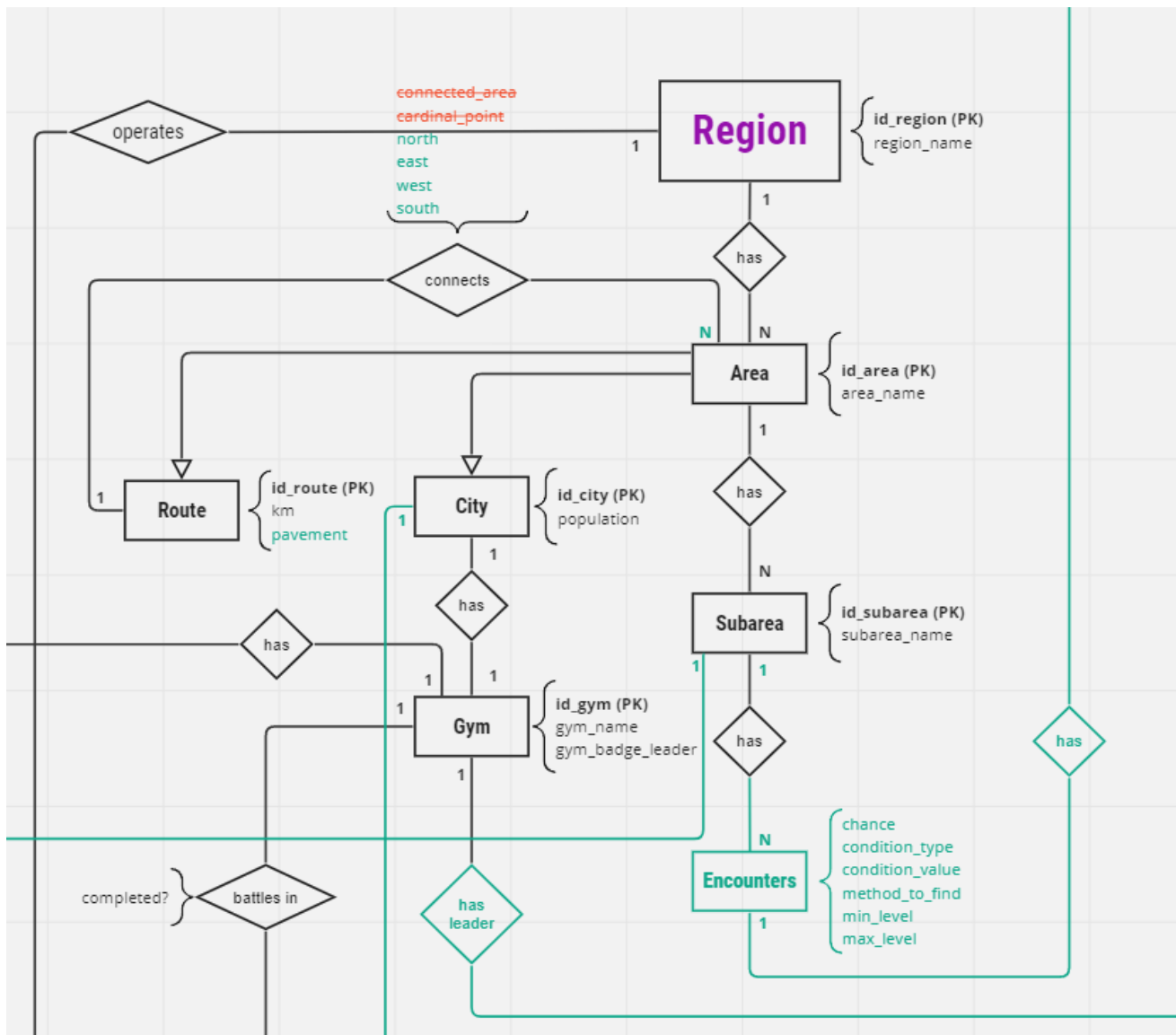
En el mòdul exploració s'han modificat les següents relacions:

- **City has Store (1:N):** S'ha afegit aquesta relació, ja que era necessària i abans no existia, degut als CSV. Aquesta relació significa les tendes que té una ciutat.
- **Pokemon_Caught has Subarea (1:1):** S'ha afegit aquesta relació, ja que era necessària segons el CSV corresponent. Aquesta relació significa a quina subàrea aquest Pokémon ha estat capturat.
- **Gym has leader Trainer (1:1):** S'ha afegit aquesta relació, ja que era necessària segons el CSV corresponent. Aquesta relació es duu a terme després de l'eliminació de l'entitat “Gym_Leader” perquè no era necessària i feia el disseny més complex. Aquesta relació significa que un gimnàs té una líder que és un entrenador.
- **Subarea has Encounters (1:N):** S'ha afegit aquesta relació, ja que era necessària segons el CSV corresponent. Aquesta relació es duu a terme després d'eliminar la relació N:M de Subarea has Pokemon com que no era correcte i va ser un error de comprensió lectora de

l'enunciat. Aquesta relació significa que en una subàrea hi ha diferents trobades de Pokémons (següent relació).

- **Encounters has Pokemon (1:1):** S'ha afegit aquesta relació, ja que era necessària segons el CSV corresponent. Aquesta relació es duu a terme després d'eliminar la relació N:M de Subarea has Pokemon perquè no era correcte i va ser un error de comprensió lectora de l'enunciat. Aquesta relació significa que en una trobada hi ha un Pokémon.

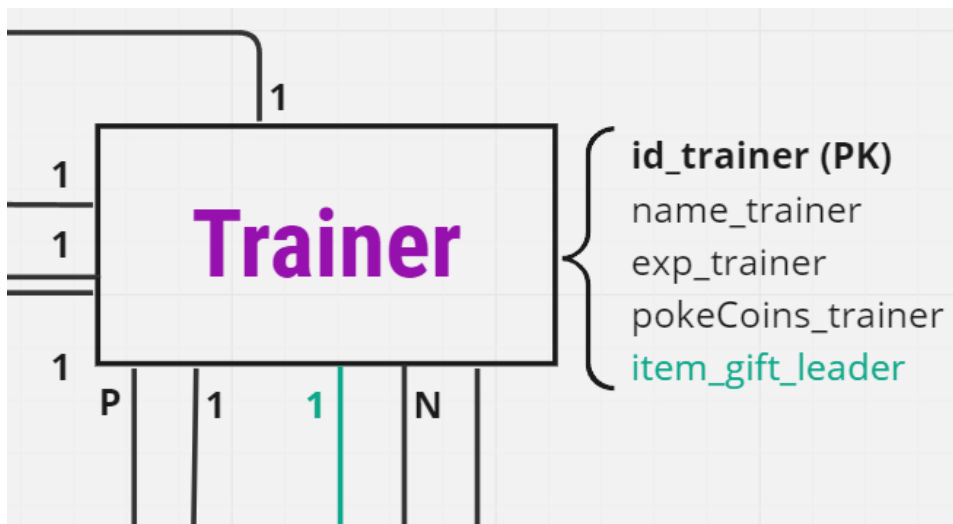
- **Route connects Area (1:N):** S'ha afegit aquesta relació, ja que era necessària segons el CSV corresponent. Aquesta relació té un canvi d'atributs:
 - **connected_area:** S'ha esborrat aquest atribut pel fet que la idea inicial era que la ruta només connectava amb una àrea.
 - **cardinal_point:** S'ha esborrat aquest atribut a causa del fet que la idea inicial era que la ruta només connectava amb una àrea i aquest atribut en concret deia per quin punt cardinal (nord, sud, est o oest) connectava a l'altra àrea.
 - **north:** S'ha afegit aquest atribut a causa del fet que la ruta pot connectar per qualsevol dels quatre punts cardinals, en aquest cas el nord, amb altra àrea.
 - **east:** S'ha afegit aquest atribut a causa del fet que la ruta pot connectar per qualsevol dels quatre punts cardinals, en aquest cas el nord, amb altra àrea.
 - **west:** S'ha afegit aquest atribut, ja que la ruta pot connectar per qualsevol dels quatre punts cardinals, en aquest cas l'est, amb altra àrea.
 - **south:** S'ha afegit aquest atribut pel fet que la ruta pot connectar per qualsevol dels quatre punts cardinals, en aquest cas el sud, amb altra àrea.
 - **pavement:** S'ha afegit aquest atribut a causa del fet que al CSV "routes" que és el que s'utilitza per inserir les dades a la taula "Route" existeix aquest camp. Aquest guarda el tipus de paviment de la ruta.



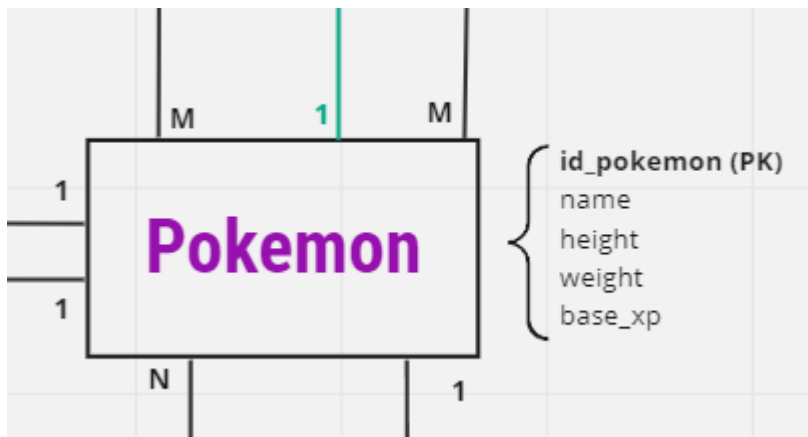
City has Store (1:N):



Gym has leader Trainer (1:1):



Encounters *has* Pokemon (1:1):



3 Actualització del model relacional

3.1 Mòdul Pokémons

Els diferents canvis que s'han hagut de fer respecte la primera entrega, han quedat reflectits en la segona entrega de la fase 1. És per això que es recomana mirar la memòria de la reentrega de la fase 1 per observar els canvis i explicacions.

3.2 Mòdul Entrenador

En el mòdul entrenador s'han modificat les següents taules (respecte la entrega sobre 10! A la entrega sobre 8 aquests canvis ja venen inclosos):

- **Trainer:**
 - **class_trainer:** S'ha afegit aquest atribut per guardar la classe de l'entrenador que ens passen en el fitxer "csv." (Fisher, Super nerd, Gym Leader...)
 - **phrase_trainer:** S'ha afegit aquest atribut per guardar la frase de l'entrenador que ens passen en el fitxer "csv."
 - **item_gift_leader:** S'ha afegit aquest atribut per guardar el ítem que regalen els entrenadors que són líders de gimnàs.
- **Gym Leader:** Aquesta taula ha estat absorbida per les taules Trainer i Gym.
- **Evil trainer:**
 - **money_stolen:** S'ha afegit aquest atribut per guardar el número de monedes que et robarà l'entrenador malvat en cas que perdis
- **Organization:**
 - **id_trainer:** S'ha esborrat aquest atribut perquè no és necessari
 - **id_partner(trainerID):** S'ha esborrat aquest atribut perquè no és necessari
 - **building_organization:** S'ha creat un camp per guardar el edifici on hi ha la organització.
 - **leader_name:** S'ha creat un camp per guardar el nom del líder de la organització.
 - **headquarter:** S'ha creat un camp per guardar el centre d'operacions de la organització.
- **Team:**
 - **id_team (PK):** S'ha creat aquest camp per poder identificar els equips de cada jugador de forma independent.
 - **id_trainer (FK):** S'ha modificat aquest camp per relacionar equip amb l'identificador del entrenador.
 - **id_pokemon_caught (FK):** S'ha creat aquest atribut per relacionar equip amb els pokemons de l'entrenador.
 - **slot:** S'ha creat aquest camp per identificar a quin slot pertany el pokemon en aquell equip.

- **pokemon(1..6)_team**: S'han esborrat tots aquests atributs per no mantenir la norma d'atomicitat en la taula.
- **Pokemon Caught**:
 - **id_pokemon_caught (PK)**: S'ha modificat aquest atribut per a que sigui totalment únic i independent.
 - **id_trainer (FK)**: S'ha modificat aquest atribut per que únicament tingui la funció de relacionar pokemon_caught amb entrenador.
 - **id_subarea (FK)**: S'ha modificat aquest atribut per que únicament tingui la funció de relacionar pokemon_caught amb subàrea.
 - **gender**: S'ha creat aquest camp per poder guardar el gènere del Pokémon.
 - **id_move 1..4**: S'han creat aquests camps per poder afegir les id dels moviments dels Pokémon capturats.
- **Battle**:
 - **id_battle (PK)**: S'ha creat aquesta primary key per poder crear la taula Battle_Statistics.
 - **experience_battle**: S'ha creat aquest atribut per poder guardar la experiència obtinguda per part d'ambdós entrenadors.
- **Battle Statistics**: S'ha creat aquesta taula per guardar les estadístiques del combat
 - **id_pokemon_caught (PK/FK)**: Es una PK que relaciona pokemon_caught.
 - **id_battle (PK/FK)**: Es una PK que relaciona battle.
 - **damage_recived**: És un atribut que guarda el mal rebut.
 - **damage_recived**: És un atribut que guarda el mal produït.
- **Contains Item**:
 - **date_time**: S'ha creat aquest camp per guardar-se la data d'obtenció d'aquest ítem.

3.3 Mòdul Objectes

Els diferents canvis que s'han hagut de fer respecte la primera entrega, han quedat reflectits en la segona entrega de la fase 1. És per això que es recomana mirar la memòria de la reentrega de la fase 1 per observar els canvis i explicacions.

3.4 Mòdul Exploració

En el mòdul exploració s'han modificat les següents taules (respecte la entrega sobre 10! A la entrega sobre 8 aquests canvis ja venen inclosos):

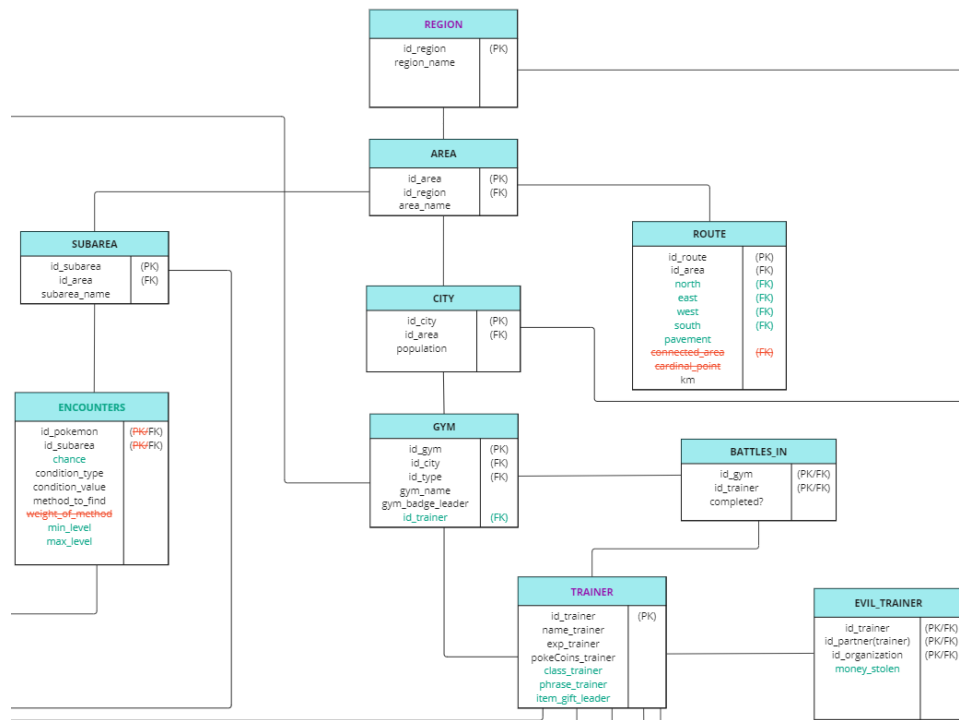
- **Route:**
 - **connected_area (FK):** S'ha esborrat aquest atribut pel fet que la idea inicial era que la ruta només connectava amb una àrea i aquesta guardava l'ID de l'àrea a la qual es connectava.
 - **cardinal_point:** S'ha esborrat aquest atribut a causa del fet que la idea inicial era que la ruta només connectava amb una àrea i aquest atribut en concret deia per quin punt cardinal (nord, sud, est o oest) connectava a l'altra àrea.
 - **north:** S'ha afegit aquest atribut a causa del fet que la ruta pot connectar per qualsevol dels quatre punts cardinals, en aquest cas el nord, amb altra àrea.
 - **east:** S'ha afegit aquest atribut a causa del fet que la ruta pot connectar per qualsevol dels quatre punts cardinals, en aquest cas el nord, amb altra àrea.
 - **west:** S'ha afegit aquest atribut a causa del fet que la ruta pot connectar per qualsevol dels quatre punts cardinals, en aquest cas l'est, amb altra àrea.
 - **south:** S'ha afegit aquest atribut a causa del fet que la ruta pot connectar per qualsevol dels quatre punts cardinals, en aquest cas el sud, amb altra àrea.
 - **pavement:** S'ha afegit aquest atribut a causa del fet que al CSV "routes" que és el que s'utilitza per inserir les dades a la taula "Route" existeix aquest camp. Aquest guarda el tipus de paviment de la ruta.

- **Gym:**
 - **id_trainer (FK):** S'ha afegit aquest atribut que és una FK de l'ID de trainer que ens dirà quin entrenador és el líder del gimnàs (s'utilitzava la taula Gym_Leader per això, però ha sigut absorbida per la taula Gym i Trainer).

- **Encounters:** S'ha canviat el nom de la taula (Pokemon_In_Area abans).
 - **id_pokemon (FK):** Aquest atribut abans era un PK/FK, però ens vam adonar que aquesta taula no era una relació N:M, per tant, el PK no feia falta, llavors s'ha deixat com a un FK que agafa l'ID de la taula Pokemon.
 - **id_subarea (FK):** Aquest atribut abans era un PK/FK, però ens vam adonar que aquesta taula no era una relació N:M, per tant, el PK no feia falta, llavors s'ha deixat com a un FK que agafa l'ID de la taula Subarea.
 - **chance:** S'ha afegit aquest atribut que ens dona la probabilitat en percentatge de què aparegui aquell pokemon. S'ha afegit a causa del fet que apareix en el CSV "encounters".
 - **condition_type:** Atribut que prèviament existia a la relació, però s'ha canviat de nom. Aquesta representa el tipus de condició perquè aparegui el pokemon.
 - **condition_value:** Atribut que prèviament existia a la relació, però s'ha canviat de nom. Aquesta representa el valor de condició perquè aparegui el Pokémon.
 - **method_to_find:** Atribut que prèviament existia a la relació, però s'ha canviat de nom. Aquesta representa el tipus de mètode perquè aparegui el pokemon.
 - **min_level:** S'ha afegit aquest atribut que ens dona el nivell mínim amb el qual pot aparèixer el pokemon. S'ha afegit a causa del fet que apareix en el CSV "encounters".

- **max_level:** S'ha afegit aquest atribut que ens dona el nivell màxim amb el que pot aparèixer el pokemon. S'ha afegit degut a que apareix en el CSV “encounters”.
- **weight_of_method:** S'ha eliminat aquest atribut ja que no apareix en el CSV “encounters” i no té sentit.

NOTA: Les taules “TRAINER”, “BATTLES_IN” i “EVIL_TRAINER” no pertanyen al mòdul exploració



4 Selecció del tipus de dades (1-2 pàgines)

- **Trainer:**
 - **id_trainer** *SERIAL*: És PK única i irrepetible per tant Serial.
 - **name_trainer** *VARCHAR*: Serà una cadena de caràcters, és un nom.
 - **exp_trainer** *INTEGER*: Serà un enter, el valor de l'experiència que tingui el Pokémon.
 - **pokeCoins_trainer** *INTEGER*: Serà el nombre de monedes d'un entrenador.
 - **class_trainer** *VARCHAR*: Serà una cadena de caràcters que indica el tipus d'entrenador.
 - **phrase_trainer** *VARCHAR*: Serà una frase per tant ampliïm de 50 a 200 caràcters.
 - **item_gift_leader** *VARCHAR*: Sol ser una Màquina Tècnica i per tant un nom.
- **Backpack:**
 - **id_backpack** *SERIAL*: És una ID PK, per tant, Serial
 - **id_trainer** *INTEGER*: És una ID que la assignem de manera manual, tan és que sigui enter que serial.
- **Battle:**
 - **id_battle** *SERIAL*: És una ID PK, per tant, Serial
 - **trainer_winner** *INTEGER*: És una ID que la assignem de manera manual, tan és que sigui enter que serial.
 - **trainer_loser** *INTEGER*: És una ID que la assignem de manera manual, tan és que sigui enter que serial.
 - **pokeCoins_battle** *INTEGER*: És una quantitat de monedes, per tant, enter.
 - **inicial_date_battle** *DATE*: És una data amb tots els camps de la data.
 - **experience_battle** *INTEGER*: És una quantitat de experiència, per tant, enter.
 - **duration_battle** *INTEGER*: És una quantitat de temps expressat en minuts, per tant, enter.
- **Region:**
 - **id_region** *SERIAL*: PK que es té que crear, per tant serial ja que aquesta va creant ID's nous.
 - **region_name** *VARCHAR*: És el nom de la regió, per tant ha de ser una cadena de caràcters.
- **Organization:**
 - **id_organization** *SERIAL*: És una PK única per tant Serial
 - **id_region** *INTEGER*: És una ID que la assignem de manera manual, tan és que sigui enter que serial.
 - **name_organization** *VARCHAR*: És un nom, cadena de caràcters
 - **building_organization** *VARCHAR*: És un camp on es troba un nom per tant cadena de caràcters.
 - **leader_name** *VARCHAR*: És el nom del líder malèvol, per tant, cadena de caràcters
 - **headquarter** *VARCHAR*: És el nom de la oficina principal, per tant, cadena de caràcters.
- **Evil Trainer:**
 - **id_trainer** *INTEGER*: És una ID que la assignem de manera manual, tan és que sigui enter que serial.
 - **id_partner** *INTEGER*: És una ID que la assignem de manera manual, tan és que sigui enter que serial.

- **id_organization** *INTEGER*: És una ID que la assignem de manera manual, tan és que sigui enter que serial.
- **money_stolen** *INTEGER*: És un nombre enter, que representa monedes robades per l'entrenador malèvol.
- **Area:**
 - **id_area** *SERIAL*: PK que es té que crear, per tant, serial per que creï les ID's.
 - **id_region** *SERIAL*: És una FK que la importem de la taula "Region", per tant es serial, pero també pot ser enter, però seguim el criteri d'utilitzar serials per les ID's.
 - **area_name** *VARCHAR*: És el nom de l'àrea, per tant ha de ser una cadena de caràcters.
- **Subarea:**
 - **id_subarea** *SERIAL*: És una PK única, s'utilitza serial però també pot ser integer perquè aquesta ID es importada des de un csv.
 - **id_area** *SERIAL*: És una FK que la importem de la taula "Area", per tant es serial, però també pot ser enter, però seguim el criteri d'utilitzar serials per les ID's
 - **subarea_name** *VARCHAR*: És el nom de la subàrea, per tant ha de ser una cadena de caràcters.
- **Pokemon Caught:**
 - **id_pokemon_caught** *SERIAL*: És una PK única, per tant, serial.
 - **id_trainer** *INTEGER*: És una ID que l'assignem de manera manual, tan és que sigui enter que serial.
 - **nick_name** *VARCHAR*: És un nom, i per tant cadena de caràcters,
 - **xp** *INTEGER*: És un nombre d'experiències en concret, per tant enter.
 - **level** *INTEGER*: És un nivell de l'1 al 100, enter.
 - **id_subarea** *INTEGER*: És una ID que l'assignem de manera manual, tant és que sigui enter que serial.
 - **date_caught** *DATE*: És la data de captura, per tant tipus date.
 - **pokeball_used_caught** *VARCHAR*: És el nom de la pokèball, cadena de caràcters.
 - **obtaining_method** *VARCHAR*: És el nom del mètode d'obtenció, cadena de caràcters
 - **pokemon_status_condition** *VARCHAR*: La status condition, cadena de caràcters
 - **pokemon_max_HP** *INTEGER*: La
 - **pokemon_remaining_HP** *INTEGER*:
 - **id_pokemon** *INTEGER*: És una ID que l'assignem de manera manual, tant és que sigui enter que serial.
 - **id_nature** *INTEGER*: És una ID que l'assignem de manera manual, tan és que sigui enter que serial.
 - **id_pokemon_ability** *INTEGER*: És una ID que l'assignem de manera manual, tant és que sigui enter que serial.
 - **id_object** *INTEGER*: És una ID que l'assignem de manera manual, tant és que sigui enter que serial.
- **Buys:**
 - **id_trainer** *INTEGER*: FK id del trainer que compra a una botiga
 - **id_store** *SERIAL*: FK id de la botiga que hi compra un trainer
 - **id_object** *SERIAL*: FK id del objecte que és comprat
- **Battle Stats:**
 - **id_pokemon_caught** *INTEGER*: FK id del pokemon que lluita

- **id_battle** *INTEGER*: FK id de la batalla
- **damage_recived** *INTEGER*: quantitat de mal rebuda
- **damage_dealt** *INTEGER*: quantitat de mal realitzada
- **Contains Item:**
 - **id_backpack** *SERIAL*: FK id de la bossa que conté l'item
 - **id_object** *SERIAL*: FK id del objecte
 - **obtention_method** *VARCHAR*: mètode d'obtenció de l'objecte
 - **date_time** *DATE*: data d'obtenció de l'objecte
- **Gym:**
 - **id_gym** *SERIAL*: PK del gimnàs, aquesta ha de ser serial perquè creï les ID's.
 - **id_city** *SERIAL*: És una FK que la importem de la taula "City", per tant, és serial, però també pot ser enter, però seguim el criteri d'utilitzar serials per les ID's.
 - **id_type** *SERIAL*: És una FK que la importem de la taula "Type_Pokemon", per tant, és serial, però també pot ser enter, però seguim el criteri d'utilitzar serials per les ID's.
 - **id_trainer** *SERIAL*: És una FK que la importem de la taula "Trainer", per tant, és serial, però també pot ser enter, però seguim el criteri d'utilitzar serials per les ID's.
 - **gym_name** *VARCHAR*: És el nom del gimnàs, per tat ha de ser una cadena de caràcters.
 - **gym_badge_leader** *VARCHAR*: És el nom de la medalla del líder del gimnàs, per tant, ha de ser una cadena de caràcters.
- **Battle in:**
 - **id_gym** *INTEGER*: PK/FK que ve donada a l'id del gimnàs
 - **id_trainer** *INTEGER*: PK/FK que ve donada a l'id de l'entrenador
 - **completed** *BOOLEAN*: Indica si aquest gimnàs ha estat completat per l'entrenador o no.
- **Team:**
 - **id_team** *SERIAL*: PK creat de manera interna per cada camp de la taula equip
 - **id_trainer** *INTEGER*: És una FK que la importem de la taula "Trainer"
 - **id_pokemon_caught** *INTEGER*: És una FK que la importem de la taula "Pokemon Caught"
 - **slot** *INTEGER*: És una FK que l'importem dels fitxers csv per saber a quin slot pertany el Pokémon.
- **City:**
 - **id_city** *SERIAL*: PK de la ciutat, aquesta ha de ser serial perquè creï les ID's.
 - **id_area** *SERIAL*: És una FK que la importem de la taula "Area", per tant, és serial, però també pot ser enter, però seguim el criteri d'utilitzar serials per les ID's.
- **Route:**
 - **id_route** *SERIAL*: PK de la ruta, aquesta ha de ser serial perquè creï les ID's.
 - **id_area** *SERIAL*: És una FK que la importem de la taula "Area", per tant, és serial, però també pot ser enter, però seguim el criteri d'utilitzar serials per les ID's.
 - **north** *INTEGER*: És una FK que la importem de la taula "Area", com pot ser null, li assignem integer.
 - **east** *INTEGER*: És una FK que la importem de la taula "Area", com pot ser null, li assignem integer.

- **west** *INTEGER*: És una FK que la importem de la taula “Area”, com pot ser null, li assignem integer.
- **south** *INTEGER*: És una FK que la importem de la taula “Area”, com pot ser null, li assignem integer.
- **pavement** *VARCHAR*: Nom del tipus de paviment de la ruta, per tant, ha de ser una cadena de caràcters.
- **km** *DECIMAL*: Guarda la distància en quilòmetres de la ruta, per tant, ha de ser un decimal perquè les dècimes no es poden menysprear.
- **Encounters:**
 - **id_pokemon** *SERIAL*: FK que la importem de la taula “Pokemon”, pot ser serial o enter, però seguint el criteri d'utilitzar serials per les ID's.
 - **id_subarea** *SERIAL*: FK que la importem de la taula “Subarea”, pot ser serial o enter, però seguim el criteri d'utilitzar serial per les ID's.
 - **method_to_find** *VARCHAR*: Nom del mètode per trobar el pokemon de la subarea, com es un nom, ha de ser una cadena de caràcters.
 - **condition_type** *VARCHAR*: Nom del tipus de condició perquè aparegui el pokemon, com és un nom, ha de ser una cadena de caràcters.
 - **condition_value** *VARCHAR*: Nom del valor de la condició perquè aparegui el pokemon, com és un nom, ha de ser una cadena de caràcters.
 - **chance** *INTEGER*: Probabilitat de què aparegui el pokemon, com és un valor en percentatge i al csv coincideix amb un enter, s'utilitza un integer.
 - **min_level** *INTEGER*: Nivell mínim del pokemon que pot aparèixer, aquest és un valor numèric i no pot tenir decimals, per tant, és un integer.
 - **max_level** *INTEGER*: Nivell màxim del pokemon que pot aparèixer, aquest és un valor numèric i ni pot tenir decimals, per tant, és un integer.
- **Growth rate:**
 - **id_pk** *SERIAL*: PK del rati del creixement, serial per anar incrementant
 - **id_growth_rate** *SERIAL*: numero de rati de creixement
 - **name_growth_rate** *VARCHAR*: Nom del rati de creixement
 - **formula** *VARCHAR*: Formula per calcular el rati de creixement
 - **level_pokemon** *INTEGER*: Valor de nivell d'un pokemon
 - **xp** *INTEGER*: Experiència per pujar a aquell nivell en concret
- **Type pokemon:**
 - **id_type** *SERIAL*: PK del tipus de pokemon, serial per anar incrementant
 - **name_type** *VARCHAR*: nom del tipus
- **Type pokemon:**
 - **id_relation** *SERIAL*: PK per identificar cada relació de mal entre tipus, serial per anar incrementant
 - **id_type_attacker** *INTEGER*: FK que es relaciona amb la taula TYPE_POKEMON on es guarda el id del tipus atacant.
 - **id_type_defender** *INTEGER*: FK que es relaciona amb la taula TYPE_POKEMON on es guarda el id del tipus defensor.
 - **Multiplier** *DECIMAL* : Relació de mal entre el tipus atacant i el defensor
- **Pokemon:**
 - **id_pokemon** *SERIAL*: PK per identificar cada espècie Pokémon
 - **name_pokemon** *VARCHAR*: Nom de l'espècie Pokémon

- **height_pokemon** *INTEGER*: Alçada de l'espècie Pokémon
- **weight_pokemon** *INTEGER*: Pes de l'espècie Pokémon
- **dex_order** *INTEGER*: Ordre en la Pokédex
- **base_xp_pokemon** *INTEGER*: Experiència base de l'espècie Pokémon
- **id_growth_rate** *INTEGER*: FK que relaciona una espècie Pokémon amb el seu rati de creixement.
- **id_type_1** *INTEGER*: FK que relaciona la espècie Pokémon amb el seu primer tipus
- **id_type_2** *INTEGER*: FK que relaciona la espècie Pokémon amb el seu segon tipus
- **Base stat:**
 - **id_id_base_stat** *SERIAL*: PK per identificar cada stat base que existeix
 - **name** *VARCHAR*: Nom del stat base
- **Acquire base stat:**
 - **id_base_stat** *INTEGER*: FK per relacionar amb el stat base
 - **id_pokemon** *INTEGER*: FK per relacionar amb l'espècie Pokémon que posseeix aquest stat base
 - **base_value** *INTEGER*: Valor del stat base en aquella espècie en concret
 - **effort** *INTEGER*: Valor d'esforç en aquella espècie en concret
- **Ability:**
 - **id_ability** *SERIAL*: PK per identificar cada habilitat
 - **name_ability** *VARCHAR*: Nom de l'habilitat
 - **full_description** *VARCHAR*: Descripció en format llarg de l'habilitat
 - **short_description** *VARCHAR*: Descripció en format curt de l'habilitat
- **Ability pokemon:**
 - **id** *SERIAL*: PK per identificar cada habilitat d'una espècie Pokémon
 - **id_ability** *INTEGER*: FK que relaciona la habilitat amb l'espècie Pokémon
 - **id_pokemon** *INTEGER*: FK que relaciona la habilitat amb l'espècie Pokémon
 - **slot** *INTEGER*: espai que ocupa l'habilitat en el conjunt de espais de l'espècie Pokémon (entre 1 i 3)
 - **is_hidden** *BOOLEAN*: indica si l'habilitat és oculta o no
- **Nature:**
 - **id_nature** *SERIAL*: PK per identificar cada natura
 - **name_nature** *VARCHAR*: Nom de la natura
 - **id_id_stat_incremented** *INTEGER*: FK que relaciona la natura amb quin stat incrementa
 - **id_id_stat_decremented** *INTEGER*: FK que relaciona la natura amb quin stat decrementa
 - **id_flavour_like** *INTEGER*: FK que relaciona la natura amb quin sabor li agrada
 - **id_flavour_dislike** *INTEGER*: FK que relaciona la natura amb quin sabor li desagrada
- **Pokemon move:**
 - **id_move** *SERIAL*: PK per identificar cada moviment
 - **name_move** *VARCHAR*: Nom del moviment
 - **accuracy** *INTEGER*: Precisió del moviment
 - **effect** *VARCHAR*: Descripció de l'efecte del moviment
 - **pp** *INTEGER*: Nombre de vegades que es pot utilitzar un moviment

- **priority_move:** *INTEGER*: Valor de prioritat del moviment
- **range:** *VARCHAR*: objectius del moviment
- **type_pokemon:** *INTEGER*: FK que relaciona el tipus amb el moviment
- **damage_type :** *VARCHAR*: Tipus de mal que fa el moviment
- **hp_healing :** *INTEGER*: Valor de curació que provoca el moviment
- **hp_drain:** *INTEGER*: Valor de absorció de vida que provoca el moviment
- **basepower_move:** *INTEGER*: Valor de potència base del moviment
- **flinch_chance:** *INTEGER*: Probabilitat de fallar
- **min_hits:** *INTEGER*: Nombre mínims de cops que efectua el moviment
- **max_hits:** *INTEGER*: Valor de absorció de vida que provoca el moviment
- **ailment:** *VARCHAR*: tipus de aliment
- **ailment_chance:** *INTEGER*: Probabilitat de provocar un aliment
- **status_move:** *VARCHAR*: tipus de efecte secundari
- **stat_change_rate:** *INTEGER*: Probabilitat de provocar un efecte secundari
- **change_amount:** *INTEGER*: valor de canvi
- **Evolves:**
 - **id_evolve** *SERIAL*: id de la evolució,
 - **id_pokemon_base** *INTEGER*: FK que relaciona la preevolució amb un pokemon
 - **id_pokemon_final** *INTEGER* FK que relaciona la postevolució amb un pokemon
 - **baby** *BOOLEAN* : indica si és bebè la preevolució
 - **method_initial** *VARCHAR* : mètode de evolució
 - **gender** *VARCHAR* : nom del gènere per evolucionar
 - **min_level** *INTEGER* : level per evolucionar
 - **time_of_day** *VARCHAR* : moment del dia per evolucionar
 - **localization** *VARCHAR* : localització per evolucionar
 - **id_item** *INTEGER* FK: id del item per evolucionar
 - **id_know_move** *INTEGER* FK: id del moviment per evolucionar
 - **min_happiness** *INTEGER* : amistat per evolucionar
- **Item:**
 - **id_object** *SERIAL PRIMARY KEY*: id de l'objecte
 - **name_object** *VARCHAR* : nom de l'objecte
 - **base_price_object** *INTEGER*: preu de compra de l'objecte
 - **description_object** *VARCHAR*: descripció de l'objecte
 - **quick_sell_price** *INTEGER*: preu de venda de l'objecte
- **Treasure:**
 - **id_treasure** *SERIAL* id del tresor
 - **collector_price** *INTEGER* preu al que el col·leccionista ven un tresor
- **Heal:**
 - **id_heal** *SERIAL* (id_object): id de l'objecte de curació
 - **heal_points** *INTEGER*: punts de curació que otorga l'objecte
 - **can_revive** *BOOLEAN*: booleà que indica si l'objecte permet al Pokemon reviure
- **Pokeball:**
 - **id_pokeball** *SERIAL*: id de la pokeball
 - **max_ratium** *INTEGER*: rati màxim al que es pot capturar
 - **min_ratium** *INTEGER*: rati mínim al que es pot capturar

- **Berry:**
 - **id_berry** *SERIAL* PRIMARY KEY REFERENCES ITEM: (id_object) És una PK única i irrepètible per la taula “Berry”
 - **name** *VARCHAR* (50): És el nom de la baya
 - **growth_time** *INTEGER*: És el nombre de temps que tarda la baya en créixer
 - **size_berry** *INTEGER*: És el nombre que indica el tamany de la baya
 - **smoothness** *VARCHAR* (50): És el nom del sabor de la baya
 - **firmness** *VARCHAR* (50): És un nom atribut de la baya, la fermesa d’aquesta
 - **max_num_harvest** *INTEGER*: És el número que indica quantes bayes es recullen a la collita,
 - **natural_gift_powder** *INTEGER*: Enter que diu el poder de la baya
 - **soil_dryness** *INTEGER*: És un enter que diu quant seca el terra d’aquesta bata

- **Flavour:**
 - **id_flavour** *SERIAL*: id del sabor
 - **name** *VARCHAR* : nom del sabor

- **Berry flavour:**
 - **id** *SERIAL* : id per saber el sabor que té una berry
 - **id_flavour** *INTEGER* : FK id del sabor de la berry
 - **id_berry** *INTEGER* : FK id de la Berry que té sabor
 - **boost** *INTEGER*: potència del sabor

- **Booster:**
 - **id_booster** *INTEGER* ITEM (id_object): id per identificar el potenciador PK/FK
 - **id_stat** *INTEGER*: id del stat que potencia FK
 - **stat_increase_time** *INTEGER*: temps que potencia l’stat

- **Sells:**
 - **id_store** *SERIAL*: id de la botiga on compra PK/FK
 - **id_object** *SERIAL*: id de l’objecte que compra FK
 - **stock** *INTEGER*: stock que hi ha a la botiga
 - **discount** *INTEGER*: descompte que s’aplica al comprar l’objecte

- **Store:**
 - **id_store** *SERIAL*: id de la botiga PK
 - **id_area** *INTEGER*: id de l’area on es troba la botiga FK
 - **number_floors** *INTEGER*: número de plantes que te la botiga
 - **store_name** *VARCHAR*(50): nom de la botiga

- **Buys:**
 - **id_transaction** *SERIAL* És la id de la transacció de la compra
 - **id_trainer** *SERIAL*: FK id del trainer que compra
 - **id_store** *SERIAL* STORE(id_store),
 - **id_object** *SERIAL*: FK id de l’objecte que es compra
 - **amount** *INTEGER*: quantitat de compra
 - **cost** *INTEGER*: cost de la compra
 - **discount** *INTEGER*, : descompte al comprar
 - **date_time** *VARCHAR*: data de compra

- **TMHM**
 - **id_mt** *INTEGER* : FK id de l’objecte de mt
 - **id_move** *INTEGER* FK id del moviment que ensenya la mt

5 Codificació del model físic

El model físic al principi l'hem dividit a 1 taula per script, primer fèiem un “mockup” de com cada taula d'acord amb el model relacional i després vam anar mirant als csv's quins atributs feien falta afegir, eliminar o modificar igual que amb les taules.

A continuació vam anar fent les insercions i vam anar comprovant que el que anàvem fent era correcte.

Després vam adjuntar tots els scripts de cadascú en un script únic i vam haver d'ordenar quines taules es creen primer, ja que com utilitzem referències, no podem crear una taula amb un atribut que depèn d'una taula que no ha sigut creada, perquè invertim un temps a veure quines són les taules independents i vam anar afegint “dependència” com si fos un arbre fins que podíem executar la taula correctament.

Una vegada tots els scripts unificats, vam provar fer les importacions i veure si les relacions que hi havia entre els nostres mòduls feien el que havien de fer correctament. En cas que no es modificava el necessari fins que tot funcionés correctament.

6 Importació de la base de dades (10-12 pàgines)

Totes les insercions en general segueixen la mateixa implementació. S'ha creat una taula de còpia amb exactament els mateixos camps del fitxer “.csv” i s'han bolcat les dades allà, per després, poder-les distribuir en els fitxers “.csv” corresponents.

Exemple de la query de creació de la taula de còpia i el COPY:

```
-- Lectura del fitxer "locations.csv"
CREATE TABLE IF NOT EXISTS LOCATIONS_INSERTION(
  region VARCHAR(50),
  area VARCHAR(50),
  subarea VARCHAR(50),
  subareaID INTEGER,
  population INTEGER
);

COPY LOCATIONS_INSERTION
FROM 'C:\Users\Public\bdd_csv\locations.csv'
DELIMITER ','
CSV HEADER;
```

6.1 Abilities.csv

Primer s'ha creat la taula de copia INSERTION_ABILITIES per copiar idènticament les files i columnes del csv. Cada columna té un delimitador de ','.

Taula temporal INSERTION_ABILITIES:

- id
- name
- effect
- short_effect

I el contingut l'hem copiat a la taula ABILITIES:

- id_ability
- name_ability
- full_description
- short_description

No ha calgut fer cap canvi ni consulta a altres taules per aquesta importació:

```
INSERT INTO ABILITY(  
    id_ability,  
    name_ability,  
    full_description,  
    short_description)  
  
SELECT DISTINCT  
    id,  
    name,  
    effect, short_effect  
  
FROM  
    INSERTION_ABILITIES;
```

Amb aquesta simple consulta podem veure si s'ha importat bé les dades:

```
1 SELECT * FROM ABILITY;
```

6.2 Battle_statistics.csv

Primer s'ha creat la taula de copia: INSERTION_BATTLE_STATS. En ella abocarem les dades del CSV battle_statistics.csv que són les següents: battleID, pokemon_instanceID, remaining_hp, damage_recived i damage_dealt. Tots aquests camps estan separats per un delimitador que serà en aquest cas el símbol ','.

Ara sí, un cop hem abocat les dades a la taula de còpia traslladarem les dades a les taules fixes de la nostra base de dades. Per aquest fitxer omplirem dues taules:

La **primera taula** que omplirem serà BATTLE_STATS amb els camps

- id_pokemon_caught
- id_battle
- damage_recived
- damage_dealt)

que pertanyen als camps de la taula INSERTION_BATTLE_STATS

- *pokemon_instanceID*
- *battleID*
- damage_recieved

- damage_dealt

com que pokemon_instanceID i battleID són FK és molt important assegurar-se que existeixen aquests camps a les taules que relaciona. Per fer-ho substituïrem els camps “pokemon_instanceID” i “battleID” pels camps de les taules corresponents:

- *pokemon_instanceID* → POKEMON_CAUGHT.id_pokemon_caught
- *battleID* → BATTLE.id_battle

això ho durem a terme en la comanda INSERT que farem un INNER JOIN amb les taules “POKEMON_CAUGHT” i “BATTLE”.

Visualització d'inner join

```
INNER JOIN POKEMON_CAUGHT ON POKEMON_CAUGHT.id_pokemon_caught = INSERTION_BATTLE_STATS.pokemon_instanceID
INNER JOIN BATTLE ON BATTLE.id_battle = INSERTION_BATTLE_STATS.battleID
```

La segona taula que omplirem serà POKEMON_CAUGHT amb el camp

- pokemon_remaining_hp

que pertany al camp de la taula INSERTION_BATTLE_STATS

- remaining_hp

la taula POKEMON_CAUGHT és immensa i és per això que en aquesta memòria es veurà com s'omple amb molts fitxers “.csv”. En aquest cas omplirem només la remaining_hp de la taula. Ho farem amb una Subquery dins d'un SELECT DISTINCT que s'ha utilitzat per a la inserció en aquella taula. La Subquery tindrà la següent sintaxi.

Visualització de la Subquery

```
(SELECT max(INSERTION_BATTLE_STATS.remaining_hp) FROM INSERTION_BATTLE_STATS
where INSERTION_BATTLE_STATS.pokemon_instanceID = POKEMON_INSTANCES_INSERTION.id)
```

6.3 Battles.csv

Primer s'ha creat la taula de còpia: INSERTION_BATTLE. En ella abocarem les dades del CSV battle.csv que són les següents: battleID, winnerID, loserID, date_time, gold_reward, experience i duration. Tots aquests camps estan separats per un delimitador que serà en aquest cas el símbol ‘,’. Ara sí, un cop hem abocat les dades a la taula de còpia traslladarem les dades a les taules fixes de la nostra base de dades. Per aquest fitxer omplirem només una taula:

La taula que omplirem serà BATTLE amb els camps

- Id_battle
- Trainer_winner
- Trainer_loser
- Inicial_date_battle
- pokeCoins_batte
- experience_battle
- duration_battle

que pertanyen als camps INSERTION_BATTLE

- battleID
- winnerID
- loserID
- date_time
- gold_reward
- experience
- duration

Com que la taula té els mateixos camps que la taula d'inserció, la comanda per traslladar les dades és molt senzilla havent de fer únicament el SELECT dels camps que es volen.

Visualització de la comanda

```
INSERT INTO BATTLE (id_battle, trainer_winner, trainer_loser, inicial_date_battle, pokeCoins_battle,  
[ ][ ][ ][ ][ ] experience_battle, duration_battle)  
SELECT battleID, winnerID, loserID, date_time, gold_reward, experience, duration  
FROM INSERTION_BATTLE;
```

6.4 Berries.csv

Per importar aquest CSV, s'ha creat una taula temporal anomenada *INSERTION_BERRY* on hi hem afegit totes les dades que hi ha en el CSV tenint en compte que el delimitador era ','.

Seguidament, hem afegit aquesta informació en la taula definitiva *BERRY* on calia fer un JOIN dels noms de les berries amb els noms dels ítems de la taula *ITEM*. De la següent manera:

```
-- BERRY INSERTION
INSERT INTO BERRY (
    id_berry,
    name,
    growth_time,
    size_berry,
    smoothness,
    firmness,
    max_num_harvest,
    natural_gift_powder,
    soil_dryness)
SELECT
    ITEM.id_object,
    INSERTION_BERRY.name,
    INSERTION_BERRY.growth_time,
    INSERTION_BERRY.berry_avg_size,
    INSERTION_BERRY.smoothness,
    INSERTION_BERRY.firmness,
    INSERTION_BERRY.max_num_harvest,
    INSERTION_BERRY.natural_gift_powder,
    INSERTION_BERRY.soil_dryness
FROM
    INSERTION_BERRY
    INNER JOIN ITEM ON ITEM.name_object = INSERTION_BERRY.name;
```

6.5 Berries_flavours.csv

En el cas d'aquest CSV, es feia d'una manera molt semblant a com s'ha explicat anteriorment. Primer s'ha creat la taula temporal *INSERTION_BERRIES_FLAVOURS* i després la definitiva a la que se li ha posat el nom *BERRY_FLAVOUR*. A la temporal, hi hem copiat totes les dades del CSV i a la definitiva s'ha relacionat mitjançant 2 joins el nom del flavour de la taula flavour amb el nom d'aquest mateix de la taula temporal i el nom de la berry de la taula Berry amb el nom d'aquesta de la taula temporal. Finalment hem afegit també la potència que l'hem obtinguda de la taula temporal.

```
-- BERRY_FLAVOUR INSERTION
INSERT INTO BERRY_FLAVOUR (
    id_flavour,
    id_berry,
    boost)
SELECT
    FLAVOUR.id_flavour,
    BERRY.id_berry,
    INSERTION_BERRIES_FLAVOURS.potency
FROM
    INSERTION_BERRIES_FLAVOURS
    INNER JOIN FLAVOUR ON FLAVOUR.name = INSERTION_BERRIES_FLAVOURS.flavour
    INNER JOIN BERRY ON BERRY.name LIKE '%' || INSERTION_BERRIES_FLAVOURS.berry || '%';
```

6.6 Damage_relations.csv

Primer s'ha creat la taula temporal *DAMAGE_RELATIONS* i s'ha copiat el contingut del csv idènticament. El delimitador, en aquest cas ','.

Taula temporal DAMAGE_RELATIONS:

- id
- attacker
- defender
- multiplier

Taula on es guarden les relacions TYPE_AGAINST:

- id_type_attacker,
- id_type_defender
- multiplier

Aquí per fer la importació calia buscar el nom del tipus atacant dins la taula TYPE_POKEEMON i guardar-hi el id dins aquesta taula. Es va fer el mateix procediment pel nom del defensor i el multiplicador el vam copiar igual:

```
INSERT INTO TYPE_AGAINST(
    id_type_attacker,
    id_type_defender,
    multiplier)
SELECT
    t1.id_type,
    t2.id_type,
    SUBSTRING(DAMAGE_RELATIONS.multiplier,2)::DECIMAL
FROM
    DAMAGE_RELATIONS
INNER JOIN TYPE_POKEEMON AS t1 ON t1.name_type = DAMAGE_RELATIONS.attacker
INNER JOIN TYPE_POKEEMON AS t2 ON t2.name_type = DAMAGE_RELATIONS.defender;
```

6.7 Encounters.csv

Per fer la importació del CSV d'**Encounters** primer es va crear una taula de còpia on guardem tots els atributs que ens trobem al CSV, després fem un COPY dient-li a quina direcció es troba el fitxer ".csv" d'on ha de copiar les dades, amb el delimitador, en aquest cas ',' i indicant si el csv conté un header, en aquest cas sí.

Una vegada s'ha copiat totes les dades en la taula de còpia, procedim a fer la insertion en la taula final, en aquest cas la taula es diu Encounters.

La taula Encounters té els següents atributs:

- id_pokemon
- id_subarea
- method_to_find
- condition_type
- condition_value
- chance
- min_level
- max_level

I la taula de còpia té els següents atributs:

- pokemon
- subareaID
- method_to_find
- condition_type
- condition_value
- chance

- min_level
- max_level

En aquesta taula li guardem totes les dades del CSV i li afegim dos ID's que són FK, una de l'ID PK de la taula Subàrea i un altre ID PK, ara de la taula Pokémon.

Les consultes més importants són dos INNER JOINS on comparo l'ID de la taula Subàrea amb l'ID de la subàrea de la taula de còpia. L'altre INNER JOIN és molt similar a l'anterior, comparo l'ID de la taula Pokémon amb l'ID del Pokémon de la taula de còpia.

Això fa que se m'assigni totes les dades correctament.

Realment aquests dos INNER JOINS no fan falta, però en el remot cas que un Pokémon o una subàrea que s'intenti importar no existeixi en la BD, aquesta no s'afegirà, ja que no té molt sentit afegir unes dades que no es poden utilitzar.

Query utilitzada:

```
INSERT INTO ENCOUNTERS(
    id_pokemon,
    id_subarea,
    method_to_find,
    condition_type,
    condition_value,
    chance,
    min_level,
    max_level)
SELECT
    POKEMON.id_pokemon,
    SUBAREA.id_subarea,
    ENCOUNTERS_INSERTION.method_to_find,
    ENCOUNTERS_INSERTION.condition_type,
    ENCOUNTERS_INSERTION.condition_value,
    ENCOUNTERS_INSERTION.chance,
    ENCOUNTERS_INSERTION.min_level,
    ENCOUNTERS_INSERTION.max_level
FROM
    ENCOUNTERS_INSERTION
    INNER JOIN POKEMON ON POKEMON.name_pokemon = ENCOUNTERS_INSERTION.pokemon
    INNER JOIN SUBAREA ON SUBAREA.id_subarea = ENCOUNTERS_INSERTION.subareaID;
```

6.8 Evolutions.csv.

Primer de tot vam crear la taula temporal INSERTION_EVOLUTIONS on vam copar tot el csv, separat pel delimitador ','.

Taula temporal INSERTION_EVOLUTIONS:

```
id
baseID
evolutionID
is_baby
trigger_ie
gender
min_level
time_of_day
location_ie
item
known_move
min_happiness
```

Taula on es guardarà EVOLVES:

- id_evolve
- id_pokemon_base
- id_pokemon_final
- baby
- method_initial
- gender
- min_level
- time_of_day
- localization
- id_item
- id_know_move
- min_happiness

Per a inserir les dades, casi tots els camps eren directe, excepte el camp del id del pokemon base i del pokemon final, que els hem hagut de buscar a la taula POKEMON mitjançant un join per el seu nom. També el moviment per evolucionar hem fet un JOIN per trobat el id a la taula MOVE que corresponia al nom donat al csv, així com l'objecte evolutiu, que hem buscat el id a la taula ITEM. Aquest és la consulta:

```
INSERT INTO EVOLVES (  
    id_pokemon_base,  
    id_pokemon_final,  
    baby,  
    method_initial,  
    gender,  
    min_level,  
    time_of_day,  
    localization,  
    id_item,  
    id_know_move,  
    min_happiness)  
  
SELECT P1.id_pokemon,  
    P2.id_pokemon,  
    is_baby,  
    trigger_ie,  
    gender,  
    min_level,  
    time_of_day,  
    location_ie,  
    IM.id_object,  
    PM.id_move,  
    min_happiness  
  
FROM INSERTION_EVOLUTIONS AS IE  
INNER JOIN POKEMON AS P1 ON P1.name_pokemon = IE.baseID  
INNER JOIN POKEMON AS P2 ON P2.name_pokemon = IE.evolutionID  
LEFT JOIN ITEM AS IM ON IM.name_object = IE.item  
LEFT JOIN POKEMON_MOVE AS PM ON PM.name_move = IE.known_move
```


6.9 Growth_rates.csv

Primer s'ha creat la taula temporal INSERTION_GROWTH_RATE i s'ha copiat el contingut del csv idènticament. El delimitador, en aquest cas ',',.

La taula temporal INSERTION_GROWTH_RATE:

- id
- name
- formula
- level
- experience

La taula GROWTH RATE:

- id_pk
- id_growth_rate
- name_growth_rate
- formula
- level_pokemon
- xp

Aquí la inserció és directa:

```
INSERT INTO GROWTH_RATE (  
    id_growth_rate,  
    name_growth_rate,  
    formula,  
    level_pokemon,  
    xp)  
  
SELECT  
    id,  
    name,  
    formula,  
    level,  
    experience  
  
FROM  
    INSERTION_GROWTH_RATE;
```

6.10 Gyms.csv

Per fer la importació del csv de **Gyms** primer es va crear una taula de còpia on guardem tots els atributs que ens trobem al csv, després fem un COPY dient-li a quina direcció es troba el fitxer .csv d'on ha de copiar les dades, amb el delimitador, en aquest cas ',', i indicant si el csv conté un header, en aquest cas sí.

Una vegada s'ha copiat totes les dades en la taula de còpia, procedim a fer la inserció en la taula final, en aquest cas la taula es diu Gym.

La taula Gym té els següents atributs:

- id_gym

- id_city
- id_type
- id_trainer
- gym_name
- gym_badge_leader

I la taula de còpia els següents atributs:

- leader
- location_name
- type_name
- badge
- gym_name

En aquesta taula li guardem només els atributs “name” i “badge” del csv a causa del fet que els altres atributs (leader, location i type) ens guardem les ID’s, en aquest cas de l’entrenador que és el líder del gimnàs, de la ciutat on es troba el gimnàs i del tipus de pokemon que utilitzaran tots els entrenadors rivals.

Aquesta insertion té un total de 4 INNER JOINS claus i un WHERE, tots són necessaris per poder desar les dades correctament.

El primer INNER JOIN es fa amb la taula Àrea on comparem el nom de l’àrea en la taula Àrea amb el nom de l’àrea que ens trobem a la taula de còpia. Hem d’anar amb compte, ja que el nom de l’àrea en la taula de còpia i la taula Àrea no són iguals, hem de passar el nom de la taula de còpia a tot minúscules, ja que així estan guardades els noms en la taula Àrea. Això es fa per trobar on està el gimnàs.

El segon INNER JOIN és comparant l’ID de la taula City amb l’ID de la taula Àrea, a causa del fet que la ciutat és una àrea, però no una àrea sempre és una ciutat. Això es fa per poder assignar una ciutat al gimnàs.

El tercer INNER JOIN es fa comparant el nom de l’entrenador de la taula Trainer amb el nom del líder de la taula de còpia. Això es fa per poder assignar el líder al seu gimnàs.

L’últim INNER JOIN es fa comparant el nom del tipus de pokemon de la taula Type_Pokemon amb el nom del tipus de Pokémon de la taula de còpia. Això es fa per poder assignar el tipus de Pokémon al gimnàs.

Per últim, tenim el WHERE que és molt important, ja que a la taula Trainer existeixen entrenadors al mateix nom, per tant, necessitem diferenciar-los segons quin tipus d’entrenador són, en aquest cas ‘Gym Leader’.

Query utilitzada:

```

INSERT INTO GYM(
    id_type,
    id_city,
    id_trainer,
    gym_name,
    gym_badge_leader)
SELECT
    TYPE_POKEMON.id_type,
    CITY.id_city,
    TRAINER.id_trainer,
    GYM_INSERTION.gym_name,
    GYM_INSERTION.badge
FROM
    GYM_INSERTION
    INNER JOIN AREA ON AREA.area_name = LOWER(GYM_INSERTION.location_name)
    INNER JOIN CITY ON CITY.id_area = AREA.id_area
    INNER JOIN TRAINER ON TRAINER.name_trainer = GYM_INSERTION.leader
    INNER JOIN TYPE_POKEMON ON TYPE_POKEMON.name_type = LOWER(GYM_INSERTION.type_name)
WHERE
    TRAINER.class_trainer LIKE '%Gym Leader%';

```

6.11 Items.csv

Importar aquest fitxer no ha sigut més difícil que importar-ne cap altre però sí que ha sigut un procés més lent ja que aquest fitxer tenia molts camps. Primer hem creat la taula temporal *INSERTION_ITEM* on hi hem copiat tota la informació del csv. Després s'ha creat la taula definitiva *ITEM* on s'han copiat només els camps que conté aquesta taula. Allà hem tingut alguns problemes ja que hi havia noms que no coincidien entre el fitxer i la nostra taula. Cal mencionar que aquesta ha sigut de les primeres importacions realitzades en el mòdul de compres ja que sense aquesta taula no es pot relacionar cap altre i és per això que potser ens ha portat més temps ja que encara no dominàvem com importar dades.

Aquesta ha sigut la query utilitzada:

```

-- ITEM INSERTION
INSERT INTO ITEM (
    id_object,
    name_object,
    base_price_object,
    description_object,
    quick_sell_price)
SELECT
    id,
    name,
    cost,
    effect,
    quick_sell_price
FROM
    INSERTION_ITEM;

```

6.12 Locations.csv

Per fer la importació del csv de **Locations** primer es va crear una taula de còpia on guardem tots els atributs que ens trobem al csv, després fem un COPY dient-li a quina direcció es troba el fitxer .csv d'on ha de copiar les dades, amb el delimitador, en aquest cas ',' i indicant si el csv conté un header, en aquest cas sí.

Una vegada s'ha copiat totes les dades en la taula de còpia, procedim a fer la insercion en les taules finals, en aquest cas tenim diferents amb els seus propis atributs:

- Region
 - o id_region
 - o region_name
- Area
 - o id_area
 - o id_region
 - o area_name
- Subarea
 - o id_subarea
 - o id_area
 - o subarea_name
- City
 - o id_city
 - o id_area
 - o population

I la taula de còpia té els següents atributs:

- o region
- o area
- o subarea
- o subareaID
- o population

Comencem per Region:

Els atributs de la taula

En aquesta consulta no utilitzem res especial més enllà d'un SELECT DISTINCT. Això té a veure que al csv Locations ens trobem un total de quatre regions, totes repetides per cada àrea i subàrea que hi ha en el csv, per tant, hem d'afegir a la taula Region només una vegada cada regió, ja que no té sentit guardar regions repetides. A aquestes regions se li assigna una ID.

Query utilitzada:

```
-- REGION INSERTION
INSERT INTO REGION(region_name)
SELECT DISTINCT region
FROM LOCATIONS_INSERTION;
```

La següent taula en la llista és Àrea:

Aquesta consulta és semblant a l'anterior, però afegim altres atributs, mantenim el SELECT DISTINCT i afegim un INNER JOIN.

En aquesta taula afegim el nom de l'àrea que es troba a la taula de còpia i l'ID de la regió on està l'àrea de la taula Region.

Fem un SELECT DISTINCT perquè el csv Locations conté els noms de les regions i àrees repetides, per tant, no necessitem guardar àrees repetides (mateix nom) sempre que estiguin en la mateixa regió, en cas que tinguin el mateix nom, però se situïn en regions diferents sí que s'ha de guardar aquesta àrea.

L'INNER JOIN que utilitzem serveix per comparar el nom de la regió en la taula Region amb el nom de la regió que ens trobem a la taula de còpia. Això fa que se'ns guardi l'àrea corresponent a la regió corresponent a la taula Àrea.

A aquestes àrees se li assignen una ID.

Query utilitzada:

```
INSERT INTO AREA(  
    area_name,  
    id_region)  
SELECT DISTINCT  
    LOCATIONS_INSERTION.area,  
    REGION.id_region  
FROM  
    LOCATIONS_INSERTION  
    INNER JOIN REGION ON LOCATIONS_INSERTION.region = REGION.region_name;
```

La penúltima taula en la llista és Subàrea:

En aquesta consulta ens guardem l'ID de la subàrea, que en aquest cas ja ens donen una ID, per tant, hem d'afegir-la, l'ID de l'àrea on es troba la subàrea i el nom de la subàrea.

A la consulta s'utilitza un DISTINCT ON de l'ID de la subàrea que obtenim del csv per evitar repeticions de les subàrees, ja que no poden existir perquè són úniques.

Després es fan dos INNER JOINS i un WHERE:

El primer per saber a quina àrea pertany la subàrea fent una comparació del nom de l'àrea de la taula Àrea amb el nom de l'àrea de la taula de còpia.

El segon per saber a quina regió es troba l'àrea i la subàrea fent una comparació del nom de la regió en la taula Region amb el nom de la regió en la taula de còpia.

El WHERE s'utilitza per no guardar subàrees buides, és a dir, no guardar "espais en blanc" a la taula degut a com està estructurat el csv.

Query utilitzada:

```
INSERT INTO SUBAREA(  
    id_subarea,  
    id_area,  
    subarea_name)  
SELECT DISTINCT ON  
    (LOCATIONS_INSERTION.subareaID) subareaID,  
    AREA.id_area,  
    LOCATIONS_INSERTION.subarea  
FROM  
    LOCATIONS_INSERTION  
    INNER JOIN AREA ON AREA.area_name = LOCATIONS_INSERTION.area  
    INNER JOIN REGION ON REGION.region_name = LOCATIONS_INSERTION.region AND  
    AREA.id_region = REGION.id_region  
WHERE  
    subareaID IS NOT NULL;
```

L'última taula, City:

Aquesta consulta és especial, ja que el concepte City en aquesta pràctica és tota aquella àrea on hi hagi població.

La consulta guarda l'ID de l'àrea perquè una ciutat és un tipus d'àrea, i també ens guardem la seva població.

Les parts importants de la consulta són l'INNER JOIN i el WHERE que s'utilitza:

L'INNER JOIN relaciona el nom de l'àrea de la taula Àrea amb el nom de l'àrea amb la taula de còpia, per guardar correctament l'ID de l'àrea a la taula Ciutat i no afegir una àrea que no sigui una ciutat a la taula.

Per últim, tenim el WHERE que el que fa és guardar totes aquelles àrees que tinguin una població, com hem comentat abans, una ciutat es considera tota àrea que tingui població, per tant, una manera fàcil de guardar les àrees és comprovant si l'àrea té població, si és així, ens guardem l'ID de l'àrea en la taula City, també ens guardem la seva respectiva població.

Query utilitzada:

```
INSERT INTO CITY(  
    id_area,  
    population)  
SELECT  
    AREA.id_area,  
    LOCATIONS_INSERTION.population  
FROM  
    LOCATIONS_INSERTION  
INNER JOIN AREA ON AREA.area_name = LOCATIONS_INSERTION.area  
WHERE  
    LOCATIONS_INSERTION.population IS NOT NULL;
```

6.13 Moves.csv

Per fer la importació del csv primer es va crear una taula de còpia on guardem tots els atributs que ens trobem al csv, després fem un COPY a la taula INSERTION_MOVES dient-li a quina direcció es troba el fitxer .csv d'on ha de copiar les dades, amb el delimitador, en aquest cas ',' i indicant si el csv conté un header, en aquest cas sí.

La taula temporal INSERTION MOVES:

- move_id
- name
- accuracy
- effect
- pp
- priority
- target
- type
- move_damage_class
- hp_healing
- hp_drain
- power
- flinch_chance ,
- min_hits
- max_hits
- ailment
- ailment_chance
- stat
- stat_change_rate

- change_amount

La taula on guardarem POKEMON_MOVE:

- id_move
- name_move
- accuracy
- effect
- pp
- priority_move
- range
- type_pokemon
- damage_type
- hp_healing
- hp_drain
- basepower_move
- flinch_chance
- min_hits
- max_hits,
- ailment
- ailment_chance
- status_move
- stat_change_rate
- change_amount

Aquí l'inserció és casi directa, només queda relacionar el tipus del moviment amb la taula de TYPE_POKEMON, mitjançant el nom del tipus i hi guardarem el id del tipus:

```

    type_pokemon,
    damage_type,
    hp_healing,
    hp_drain,
    basepower_move,
    flinch_chance,
    min_hits,
    max_hits,
    ailment,
    ailment_chance,
    status_move,
    stat_change_rate,
    change_amount)
SELECT
    move_id,
    name,
    accuracy,
    effect,
    pp,
    priority,
    target,
    TYPE_POKEMON.id_type,
    move_damage_class,
    hp_healing,
    hp_drain,
    power,
    flinch_chance,
    min_hits,
    max_hits,
    ailment,
    ailment_chance,
    stat,
    stat_change_rate,
    change_amount
FROM
    INSERTION_MOVES
INNER JOIN TYPE_POKEMON ON TYPE_POKEMON.name_type = INSERTION_MOVES.type;

```

6.14 Natures.csv

Per fer la importació del csv primer es va crear una taula de còpia on guardem tots els atributs que ens trobem al csv, després fem un COPY a la taula INSERTION_NATURE dient-li a quina direcció es troba el fitxer .csv d'on ha de copiar les dades, amb el delimitador, en aquest cas ',' i indicant si el csv conté un header, en aquest cas sí.

Taula temporal INSERTION_NATURE:

- id
- name
- decreased_stat
- increased_stat
- likes_flavor
- hates_flavor

Taula on es guardarà la informació NATURE:

- id_nature
- name_nature
- id_stat_incremented
- id_stat_decremented
- id_flavour_like

- id_flavour_dislike

Per a inserir la informació han calgut 4 JOIN per poder relacionar el nom del stat que augmenta i decrementa amb el id de la taula BASE_STAT. EL mateix passava amb el sabor que agrada i desagrada, relacionàvem aquesta taula amb la taula FLAVOUR per buscar el id que tingués el nom del fitxer i hem guardat els ids a la taula NATURE. LA resta de camps són copiats directament de la taula d'inserció.

```
INSERT INTO NATURE(
    id_nature,
    name_nature,
    id_stat_incremented,
    id_stat_decremented,
    id_flavour_like,
    id_flavour_dislike)

SELECT
    INSERTION_NATURE.id,
    INSERTION_NATURE.name,
    b1.id_base_stat,
    b2.id_base_stat,
    f1.id_flavour,
    f2.id_flavour

FROM
    INSERTION_NATURE
LEFT JOIN BASE_STAT as b1 ON b1.name = INSERTION_NATURE.increased_sta
LEFT JOIN BASE_STAT as b2 ON b2.name = INSERTION_NATURE.decreased_sta
LEFT JOIN FLAVOUR as f1 ON f1.name = INSERTION_NATURE.likes_flavor
LEFT JOIN FLAVOUR as f2 ON f2.name = INSERTION_NATURE.hates_flavor;
```

6.15 Pokemon.csv

Per fer la importació del csv primer es va crear una taula de còpia on guardem tots els atributs que ens trobem al csv, després fem un COPY a la taula INSERTION_POKEMON dient-li a quina direcció es troba el fitxer .csv d'on ha de copiar les dades, amb el delimitador, en aquest cas ',' i indicant si el csv conté un header, en aquest cas sí.

Taula temporal INSERTION_POKEMON

- id_pokemon
- pokemon
- baseExperience
- height
- weight
- dex_order
- growth_rate_id
- type1
- type2

Taula on es guardarà POKEMON:

- id_pokemon
- name_pokemon
- height_pokemon
- weight_pokemon

- dex_order
- base_xp_pokemon
- id_growth_rate
- id_type_1
- id_type_2

Per importar dins la taula POKEMON, ha calgut relacionar el número de tipus de growth_rate amb la taula GROWTH_RATE i anotar el id corresponent a la taula. EL mateix passa amb els tipus, cal buscar a la taula TYPE_POKEMON el nom del dos tipus i guardar a la taula només el id del tipus 1 i del tipus 2.

```
INSERT INTO POKEMON (
    id_pokemon,
    name_pokemon,
    height_pokemon,
    weight_pokemon,
    dex_order,
    base_xp_pokemon,
    id_growth_rate,
    id_type_1,
    id_type_2)
SELECT DISTINCT ON
    (INSERTION_POKEMON.id_pokemon) id_pokemon,
    INSERTION_POKEMON.pokemon,
    INSERTION_POKEMON.height,
    INSERTION_POKEMON.weight,
    INSERTION_POKEMON.dex_order,
    INSERTION_POKEMON.baseExperience,
    GROWTH_RATE.id_pk + 1,
    t1.id_type,
    t2.id_type
FROM
    INSERTION_POKEMON
    INNER JOIN GROWTH_RATE ON GROWTH_RATE.id_growth_rate = INSERTION_POKEMON.growth_rate_ID
    INNER JOIN TYPE_POKEMON as t1 ON t1.name_type LIKE INSERTION_POKEMON.type1
    LEFT JOIN TYPE_POKEMON as t2 ON t2.name_type LIKE INSERTION_POKEMON.type2
WHERE
    GROWTH_RATE.xp < INSERTION_POKEMON.baseExperience;
```

6.16 Pokemon_abilities.csv

Primer s'ha creat la taula de còpia INSETION_POKEMON_ABILITIES En ella abocarem les dades del CSV. Tots aquests camps estan separats per un delimitador que serà en aquest cas el símbol ','. Ara sí, un cop hem abocat les dades a la taula de còpia traslladarem les dades a les taules fixes de la nostra base de dades. Per aquest fitxer omplirem només una taula:

La taula temporal INSETION_POKEMON_ABILITIES:

- id
- speciesID
- abilityID
- slot
- is_hidden

La taula on es guardarà ABILITY_POKEMON:

- id
- id_ability
- id_pokemon
- slot
- is_hidden

Per inserir les dades, calia relacionar el id de la habilitat del fitxer amb el de la taula ABILITY i guardar-hi el id. El mateix passava amb el pokemon, que havíem de guardar el id que corresponia a la taula POKEMON i a més, guardar el slot que ocupa aquesta habilitat en el pokemon:

```
INSERT INTO ABILITY_POKEMON (  
    id_ability,  
    id_pokemon,  
    slot,  
    is_hidden)  
  
SELECT  
    ABILITY.id_ability,  
    POKEMON.id_pokemon,  
    INSERTION_ABILITIES_POKEMON.slot,  
    INSERTION_ABILITIES_POKEMON.is_hidden  
  
FROM  
    INSERTION_ABILITIES_POKEMON  
INNER JOIN ABILITY ON ABILITY.id_ability = INSERTION_ABILITIES_POKEMON.abilityID  
INNER JOIN POKEMON ON POKEMON.id_pokemon = INSERTION_ABILITIES_POKEMON.speciesID;
```

6.17 *Pokemon_instances.csv*

Primer s'ha creat la taula de còpia POKEMON_INSTANCES_INSERTION. En ella abocarem les dades del CSV pokemon_instances.csv que són les següents: id, pokemon_speciesID, nickname, ownerID, level, experience, gender, nature, item, datetime, location_subareaID, obtention_method, pokeballID, move1, move2, move3 i move4. Tots aquests camps estan separats per un delimitador que serà en aquest cas el símbol ','.

Ara sí, un cop hem abocat les dades a la taula de còpia traslladarem les dades a les taules fixes de la nostra base de dades. Per aquest fitxer omplirem només una taula:

La taula que omplirem serà POKEMON_CAUGHT amb els camps

- id_pokemon_caught
- id_trainer
- nick_name
- xp
- level
- date_caught
- pokeball_used_caught
- obtaining_method
- id_pokemon
- gender
- pokemon_item
- pokemon_nature
- location_subareaID

L'omplirem amb els camps de la taula POKEMON_INSTANCES_INSERTION:

- id
- ownerID
- nickname
- experience
- level
- datetime
- pokeballID
- obtention_method
- pokemon_speciesID

- *gender*
- *item*
- *nature*
- *location_subareaID*

Quasi tots els camps són propis de la taula POKEMON_INSTANCES_INSERTION, però “item”, “nature” i “location_subareaID” són camps que a la taula POKEMON_CAUGHT té com a FK i, per tant, les insercions s’hauran de fer utilitzant INNER JOINS i/o LEFT JOINS (en el cas de les Subàreas).

6.18 Poketeams.csv

Primer s’ha creat la taula de copia INSERTION_TO_TEAMS. En ella bolcarem les dades del CSV poketeams.csv que són les següents: trainer, pokemon, slot, hp, status. Tots aquests camps estan separats per un delimitador que serà en aquest cas el símbol ‘,’.

Ara sí, un cop hem bolcat les dades a la taula de copia traslladarem les dades a les taules fixes de la nostra base de dades. Per aquest fitxer omplirem dues taules:

La primera taula que omplirem serà TEAM amb els següents camps:

- id_trainer
- id_pokemon_caught
- slot

I ho farem amb els camps de INSERTION_TO_TEAMS següents:

- trainer
- pokemon
- slot

Com tots els camps de la taula coincideixen no cal fer ús de cap JOIN ni de cap Subquery, per tant, la inserció queda d’aquesta manera:

Visualització de la Inserció

```
INSERT INTO TEAM (id_trainer, id_pokemon_caught, slot)
SELECT DISTINCT id_trainer, pokemon_number, slot
FROM INSERTION_TO_TEAMS;
```

La segona taula que omplirem serà POKEMON_CAUGHT que com bé he dit en anterior fitxer “.csv” és una taula inmensa i s’omple a poc a poc amb molts CSV. En aquest cas els camps que omplirem seran:

- pokemon_max_hp
- pokemon_status_condition

Amb els camps d’INSERTION_TO_TEAMS següents:

- hp
- status

Per tant, a la comanda d’inserció a POKEMON_CAUGHT haurem de fer un INNER JOIN amb INSERTION_TO_TEAMS per poder agafar aquells camps.

6.19 Purchases.csv

Per a importar aquest CSV, s'ha creat la taula temporal *INSERTION_PURCHASES* on s'han copiat totes les dades del fitxer *Purchases.csv* per a després copiar les que ens interessin a la taula definitiva *BUYS*. Per a relacionar la taula provisional amb la definitiva, s'han hagut de realitzar 3 joins entre elles:

```
-- BUYS insertion
INSERT INTO BUYS (
    id_trainer,
    id_store,
    id_object,
    amount,
    cost,
    discount,
    date_time)
SELECT TRAINER.id_trainer, STORE.id_store, ITEM.id_object, INSERTION_PURCHASES.amount,
INSERTION_PURCHASES.cost, INSERTION_PURCHASES.discount, INSERTION_PURCHASES.date_time
FROM INSERTION_PURCHASES
INNER JOIN TRAINER ON TRAINER.id_trainer = INSERTION_PURCHASES.trainerID
INNER JOIN STORE ON STORE.id_store = INSERTION_PURCHASES.storeID
INNER JOIN ITEM ON ITEM.id_object = INSERTION_PURCHASES.itemID;
```

Un primer join per unir el ID dels trainers (de la taula *TRAINER*) amb el ID de la taula provisional. Un segon join per unir el ID de la botiga (de la taula *STORE*) amb el ID de la taula provisional. Un últim join per unir el ID del item (de la taula *ITEM*) amb el ID de la taula provisional.

6.20 Routes.csv

NOTA: Aquest CSV té un error (no el codi proporcionat), o al menys jo el considero com erroni. S'explica tot a l'apartat 7.4.3.

Per fer la importació del csv de **Routes** primer es va crear una taula de copia on guardem tots els atributs que ens trobem al csv, després fem un COPY dient-li a quina direcció es troba el fitxer .csv d'on té que copiar les dades, amb el delimitador, en aquest cas ',', i indicant si el csv conté un header, en aquest cas si.

Una vegada s'ha copiat totes les dades en la taula de copia, procedim a fer la inserció en la taula final, en aquest cas la taula es diu Route.

Els atributs de la taula Route són els següents:

- id_route
- id_area
- north
- east
- west
- south
- pavement
- km

I els de la taula de copia:

- route
- north
- east
- west
- south

- paviment

A aquesta taula li guardem tots els atributs del csv menys els atributs “route”, “north”, “east”, “west” i “south” ja que en comptes de guardar el nom, guardarem l’ID de l’àrea, ja que un àrea pot ser una ruta, també guardarem una distància de la ruta, aquesta com en el csv no existeix, doncs la creem aleatòriament amb la funció RANDOM().

Les àrees que connecten pels punts cardinals les guardem totes en minúscules per que coincideixin amb el nom de les àrees i no hi hagi cap confusió en una possible query per buscar aquella àrea on connecta.

Aquesta insertion té un únic INNER JOIN que compara el nom de l’àrea de la taula Area amb el nom de l’àrea de la taula de copia (tota en minúscules per que els noms coincideixin) per poder guardar l’ID de l’àrea correctament i desar les dades com calen.

També té 4 LEFT JOINS que comparen el nom de l’àrea que connecta per cada punt cardinal per assignar l’ID de l’àrea correctament.

Query utilitzada:

```
INSERT INTO ROUTE(
    north,
    east,
    west,
    south,
    paviment,
    id_area,
    km)
SELECT
    north.id_area,
    east.id_area,
    west.id_area,
    south.id_area,
    ROUTE_INSERTION.paviment,
    AREA.id_area,
    RANDOM()*(100)
FROM
    ROUTE_INSERTION
    INNER JOIN AREA ON AREA.area_name = LOWER(ROUTE_INSERTION.route)
    LEFT JOIN AREA AS north ON north.area_name = LOWER(ROUTE_INSERTION.north)
    LEFT JOIN AREA AS east ON east.area_name = LOWER(ROUTE_INSERTION.east)
    LEFT JOIN AREA AS west ON west.area_name = LOWER(ROUTE_INSERTION.west)
    LEFT JOIN AREA AS south ON south.area_name = LOWER(ROUTE_INSERTION.south);
```

6.21 Stats.csv

Primer s’ha creat la taula de INSERTION_STATS. En ella abocarem les dades del CSV. Tots aquests camps estan separats per un delimitador que serà en aquest cas el símbol ‘,’.

La taula temporal INSERTION_STATS:

- id
- stat
- pokemon
- base_stat
- effort

La taula on es guardarà els diferents tipus de stats BASE_STAT:

- id_base_stat
- name

La taula on es guardarà la relació entre els stats i cada espècie pokemon ACQUIRE_BASE_STAT:

- id_base_stat
- id_pokemon
- base_value
- effort

Per a omplir la taula BASE_STAT, calia seleccionar els DISTINCT noms de stat que existien i assignar-li un id SERIAL:

```
INSERT INTO BASE_STAT(name)
SELECT DISTINCT stat
FROM INSERTION_STATS;
```

Per a omplir la taula ACQUIRE_BASE_STAT calia buscar el id del BASE_STAT que teníem com a nom en el fitxer i relacionar-los per guardar el id el BASE_STAT. Pel Pokémon passa igual, buscarem el nom en la taula POKEMON i hi guardarem el id a la nostra taula. Després copiarem el valor_base i els punts d'esforç:

```
INSERT INTO ACQUIRE_BASE_STAT (
    id_base_stat,
    id_pokemon,
    base_value,effort)
SELECT
    BASE_STAT.id_base_stat,
    POKEMON.id_pokemon,
    INSERTION_STATS.base_stat,
    INSERTION_STATS.effort
FROM
    INSERTION_STATS
    INNER JOIN POKEMON ON INSERTION_STATS.pokemon = replace(POKEMON.name_pokemon, '-', ' ')
    INNER JOIN BASE_STAT ON BASE_STAT.name = INSERTION_STATS.stat;
```

6.22 Storeitems.csv

El primer que s'ha fet ha sigut crear una taula que ens servirà de manera provisional on importarem totes les dades del fitxer CSV. Tots els camps d'aquest fitxer es troben delimitat per el símbol ','. Un cop abocades aquestes, seleccionarem aquelles que ens interessin per a copiar-les a la taula SELLS. Per a completar la taula definitiva, caldrà utilitzar un parell de joins per a relacionar la taula SELLS amb les taules TRAINER, STORE i ITEM.

En la següent captura es veu el INSERT que es realitza a la taula definitiva i els joins que ha calgut fer.

```
-- BUYS insertion
INSERT INTO BUYS (
    id_trainer,
    id_store,
    id_object,
    amount,
    cost,
    discount,
    date_time)
SELECT TRAINER.id_trainer, STORE.id_store, ITEM.id_object, INSERTION_PURCHASES.amount,
INSERTION_PURCHASES.cost, INSERTION_PURCHASES.discount, INSERTION_PURCHASES.date_time
FROM INSERTION_PURCHASES
INNER JOIN TRAINER ON TRAINER.id_trainer = INSERTION_PURCHASES.trainerID
INNER JOIN STORE ON STORE.id_store = INSERTION_PURCHASES.storeID
INNER JOIN ITEM ON ITEM.id_object = INSERTION_PURCHASES.itemID;
```

6.23 Stores.csv

El que s'ha realitzat en primer lloc ha sigut crear-nos la taula temporal per importar en ella les dades del CSV, les quals venen delimitades per ','.

Seguidament, ens hem creat la taula definitiva de *STORES* on hem afegit en cadascun dels camps les dades guardades a la taula temporal excepte el camp de l'*id_area* ja que calia relacionar la botiga amb l'àrea on estava ubicada aquesta i hem hagut d'obtenir aquesta informació de la taula *AREA*. D'aquesta manera doncs, s'ha hagut de fer un JOIN entre *AREA* i la taula temporal de *STORES*. En la següent captura es pot veure amb més detall.

```
-- STORE insertion
INSERT INTO STORE (
    id_store,
    id_area,
    number_floors,
    store_name)
SELECT INSERTION_STORES.storeID, AREA.id_area,
INSERTION_STORES.floors, INSERTION_STORES.store_name
FROM INSERTION_STORES
INNER JOIN AREA ON AREA.area_name = INSERTION_STORES.city;
```

6.24 Traineritems.csv

Primer s'ha creat la taula de còpia *TRAINER_ITEMS_INSERTION*. En ella abocarem les dades del CSV *poketeams.csv* que són les següents: *trainerID*, *itemID*, *obtention_method*, *date_time*. Tots aquests camps estan separats per un delimitador que serà en aquest cas el símbol ','.

Ara sí, un cop hem abocat les dades a la taula de còpia traslladarem les dades a les taules fixes de la nostra base de dades. Per aquest fitxer omplirem una taula:

La taula que omplirem és *CONTAINS_ITEM* amb els camps següents:

- *Id_backpack*
- *Id_object*
- *obtention_method*
- *datetime*

I ho omplirem amb els camps:

- *trainerID*
- *itemID*
- *obtention_method*
- *date*

Com aquesta taula ja agafa tots els atributs de la taula *TRAINER_ITEMS_INSERTION* no cal seguir-la utilitzant en futures insercions que es facin doncs ja es pot utilitzar directament *CONTAINS_ITEM*.

6.25 Trainers.csv

Primer s'ha creat la taula de còpia *INSERTION_TRAINERS*. En ella abocarem les dades del CSV *poketeams.csv* que són les següents: *id*, *trainer_class*, *name*, *experience*, *gold*, *gift_item*, *villain_team*, *partnerID*, *phrase*, *money_stolen*. Tots aquests camps estan separats per un delimitador que serà en aquest cas el símbol ','.

Ara sí, un cop hem abocat les dades a la taula de còpia traslladarem les dades a les taules fixes de la nostra base de dades. Per aquest fitxer omplirem dues taules:

La primera taula que omplirem serà TRAINERS:

- Id_trainer
- Name_trainer
- Exp_trainer
- pokeCoins_trainer
- class_trainer
- phrase_trainer
- item_gift_leader

L'omplirem amb els camps:

- id
- name
- experience
- gold
- class
- phrase
- gift_item

Depenent de si té o no gift_item considerarem l'entrenador com a líder de gimnàs o no, per tant, la taula de líder de gimnàs ja no ens cal. Ara bé sí que necessitem encara diferenciar si un entrenador és un entrenador malèvol, per tant, la classe EVIL_TRAINER també li afegirem dades d'aquest CSV, en particular la dada del partnerID

Quedaria així la segona taula EVIL_TRAINER amb els camps

- id_trainer
- id_partner

L'omplirem amb els camps de la taula INSERTION_TRAINER amb els camps

- id
- partnerID

6.26 Types.csv

Primer s'ha creat la taula de còpia INSERTION_TYPE per inserir les dades del csv amb delimitador de ',':

Taula temporal INSERTION_TYPE:

- id
- name

Taula on es guaradrà el TYPE_POKEMON:

- id_type
- name_type

Aquí la inserció és directa:

```

INSERT INTO TYPE_POKEMON (
    id_type,
    name_type)
SELECT
    id,
    name
FROM
    INSERTION_TYPE;

```

6.27 Villainous_organizations.csv

Primer s'ha creat la taula de còpia INSERTION_ORGANIZATION. En ella abocarem les dades del CSV poketeams.csv que són les següents: name, building , hq, leader, region. Tots aquests camps estan separats per un delimitador que serà en aquest cas el símbol ','.

Ara sí, un cop hem abocat les dades a la taula de còpia traslladarem les dades a les taules fixes de la nostra base de dades. Per aquest fitxer omplirem una:

La taula que omplirem serà ORGANIZATION amb els camps:

- *id_region*
- name_organization
- building_organization
- leader_name
- headquarter

L'omplirem amb els camps de la taula INSERTION_ORGANIZATION següents

- region
- name
- building
- leader
- region

La qüestió aquí és que region és un nom a la taula d'importació i, per tant, no ens quadra amb la id_region que tenim a la taula d'Organization, i és per això que el que s'ha fet és agafar l'id_region de la taula region que coincideixi al nom amb la taula d'inserció.

7 Validació de la base de dades

7.1 Pokémons (1-2 pàgines)

7.1.1 Conjunt de consultes:

```
--Query per buscar la formula del growth_rate associat a un pokemon, juntament amb l'experiència per pujar
--a cada nivell.

SELECT * FROM GROWTH_RATE
WHERE GROWTH_RATE.formula =
(SELECT GROWTH_RATE.formula
FROM GROWTH_RATE
INNER JOIN POKEMON ON POKEMON.id_growth_rate = GROWTH_RATE.id_pk
WHERE POKEMON.name_pokemon LIKE 'sentret');

--Query per mostrar els tipus de cada pokemon, on el primer és obligatori i el segon opcional
SELECT POKEMON.name_pokemon, type1.name_type, type2.name_type
FROM POKEMON
INNER JOIN TYPE_POKEMON AS type1 ON type1.id_type = POKEMON.id_type_1
LEFT JOIN TYPE_POKEMON AS type2 ON type2.id_type = POKEMON.id_type_2;

--Query per buscar els stats base d'una espècie pokemon en concret
SELECT stat.name , acstat.base_value, acstat.effort
FROM POKEMON
INNER JOIN ACQUIRE_BASE_STAT as acstat ON acstat.id_pokemon = POKEMON.id_pokemon
INNER JOIN BASE_STAT as stat ON stat.id_base_stat = acstat.id_base_stat
WHERE POKEMON.name_pokemon LIKE 'dratini';

--Query per mostrar el stat que incrementa i que decremента i quin sabaor li agrada i desagrada
SELECT NATURE.name_nature as nature ,b1.name as stat_increase, b2.name as stat_decrease, f1.name as flavour_like, f2.name
FROM NATURE
LEFT JOIN BASE_STAT as b1 ON b1.id_base_stat = NATURE.id_stat_incremented
LEFT JOIN BASE_STAT as b2 ON b2.id_base_stat = NATURE.id_stat_decremented
LEFT JOIN FLAVOUR as f1 ON f1.id_flavour = NATURE.id_flavour_like
LEFT JOIN FLAVOUR as f2 ON f2.id_flavour = NATURE.id_flavour_dislike;

--Query per mostrar les possibles habilitats de cada pokemon
SELECT POKEMON.name_pokemon, ABILITY.name_ability , slot, is_hidden
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;

--Query per mostrar quin els multiplicadors que té un tipus en concret contra uns altres
SELECT type2.name_type as defender, TYPE_AGAINST.multiplier
FROM TYPE_AGAINST
INNER JOIN TYPE_POKEMON as type1 ON type1.id_type = TYPE_AGAINST.id_type_attacker
INNER JOIN TYPE_POKEMON as type2 ON type2.id_type = TYPE_AGAINST.id_type_defender
WHERE type1.name_type LIKE 'water'
ORDER BY TYPE_AGAINST.multiplier DESC;

--Query per mostrar tots els objectes que ensenyen un moviment i el nom d'aquest
SELECT ITEM.name_object, POKEMON_MOVE.name_move
FROM TM_HM
INNER JOIN ITEM ON ITEM.id_object = TM_HM.id_mt
INNER JOIN POKEMON_MOVE ON POKEMON_MOVE.id_move = TM_HM.id_move
ORDER BY name_object asc;
```

7.1.2 Peticions per separat:

En la primera consulta es vol buscar el growth_rate d'un pokemon en concret, així com per cada nivell quanta experiència es necessita. Per aconseguir-ho, es selecciona tots els camps de la taula growth_rate i es filtra per aquell que la formula coincideixi amb el resulta d'una subquery. Aquesta

subquery busca la formula del growth_rate associat a un pokemon que té el mateix nom que el especificat en el where de la subquery. Aquest n'és el resultat:

	id_pk [PK] integer	id_growth_rate integer	name_growth_rate character varying (50)	formula character varying (1000)	level_pokemon integer	xp integer
1	101	2	medium	x^3	1	0
2	102	2	medium	x^3	2	8
3	103	2	medium	x^3	3	27
4	104	2	medium	x^3	4	64
5	105	2	medium	x^3	5	125
6	106	2	medium	x^3	6	216
7	107	2	medium	x^3	7	343
8	108	2	medium	x^3	8	512
9	109	2	medium	x^3	9	729
10	110	2	medium	x^3	10	1000
11	111	2	medium	x^3	11	1331
12	112	2	medium	x^3	12	1728
13	113	2	medium	x^3	13	2197
14	114	2	medium	x^3	14	2744
15	115	2	medium	x^3	15	3375
16	116	2	medium	x^3	16	4096
17	117	2	medium	x^3	17	4913
18	118	2	medium	x^3	18	5832
19	119	2	medium	x^3	19	6859
20	120	2	medium	x^3	20	8000
21	121	2	medium	x^3	21	9261
22	122	2	medium	x^3	22	10648
23	123	2	medium	x^3	23	12167
24	124	2	medium	x^3	24	13824
25	125	2	medium	x^3	25	15625
26	126	2	medium	x^3	26	17576
27	127	2	medium	x^3	27	19683

La segona consulta es vol mostrar cada pokemon quins tipus li pertanyen. Per aquesta consulta es selecciona el nom del pokemon, així com el del tipus 1 i tipus 2. Es relaciona mitjançant JOIN el id del tipus 1 amb el id del tipus 1 del pokemon i, d'igual manera la de tipus 2. El resultat és el següent:

	name_pokemon character varying (50) 🔒	name_type character varying (50) 🔒	name_type character varying (50) 🔒
1	bulbasaur	grass	poison
2	ivysaur	grass	poison
3	venusaur	grass	poison
4	charmander	fire	[null]
5	charmeleon	fire	[null]
6	charizard	fire	flying
7	squirtle	water	[null]
8	wartortle	water	[null]
9	blastoise	water	[null]
10	caterpie	bug	[null]
11	metapod	bug	[null]
12	butterfree	bug	flying
13	weedle	bug	poison
14	kakuna	bug	poison
15	beedrill	bug	poison
16	pidgey	normal	flying
17	pidgeotto	normal	flying
18	pidgeot	normal	flying
19	rattata	normal	[null]
20	raticate	normal	[null]
21	spearow	normal	flying
22	fearow	normal	flying
23	ekans	poison	[null]
24	arbok	poison	[null]
25	pikachu	electric	[null]
26	raichu	electric	[null]
27	sandshrew	ground	[null]

En la tercera consulta es vol mostrar els 6 stats base que té cada espècie pokemon, donat un nom de cerca de pokemon. Es selecciona el nom del stat base, així com el valor base i esforç. Es relaciona el id del stat amb la taula que guarda els stats de cada pokemon i després es relaciona amb la taula pokemon. Finalment es filtra segons el nom del pokemon que es vol mostrar:

	name character varying (20) 🔒	base_value integer 🔒	effort integer 🔒
1	attack	64	1
2	defense	45	0
3	hp	41	0
4	special attack	50	0
5	special defense	50	0
6	speed	50	0

En la quarta consulta es vol mirar oer cada natura quin stat incrementa i decrementa, així com quin sabor li agrada i desagrada. Es selecciona el nom de la natura, del stat incrementat i decrementat i del sabor que li agrada i desagrada. Es fan JOIN per enllaçar el id de cada stats amb la taula BASE_STAT i també amb el id dels sabors a la taula FLAVOUR. Aquest és el resultat:

	nature character varying (50)	stat_increase character varying (20)	stat_decrease character varying (20)	flavour_like character varying (50)	flavour_dislike character varying (50)
1	hasty	speed	defense	sweet	sour
2	lonely	attack	defense	spicy	sour
3	mild	special attack	defense	dry	sour
4	gentle	special defense	defense	bitter	sour
5	naive	speed	special defense	sweet	bitter
6	naughty	attack	special defense	spicy	bitter
7	rash	special attack	special defense	dry	bitter
8	lax	defense	special defense	sour	bitter
9	jolly	speed	special attack	sweet	dry
10	adamant	attack	special attack	spicy	dry
11	careful	special defense	special attack	bitter	dry
12	impish	defense	special attack	sour	dry
13	timid	speed	attack	sweet	spicy
14	modest	special attack	attack	dry	spicy
15	calm	special defense	attack	bitter	spicy
16	bold	defense	attack	sour	spicy
17	brave	attack	speed	spicy	sweet
18	quiet	special attack	speed	dry	sweet
19	sassy	special defense	speed	bitter	sweet
20	relaxed	defense	speed	sour	sweet
21	hardy	[null]	[null]	[null]	[null]
22	docile	[null]	[null]	[null]	[null]
23	bashful	[null]	[null]	[null]	[null]
24	quirky	[null]	[null]	[null]	[null]
25	serious	[null]	[null]	[null]	[null]

En la cinquena consulta es vol mostrar les diferents habilitats que poden tenir els pokemons. Es selecciona el nom de cada pokemon, el nom de la habilitat, a quin slot pertany i si es tracta d'una habilitat oculta o no. Es relaciona la taula pokemon i ability en la taula POKEMON_ABILITY mitjançant JOIN pels seus id. Aquest és el resultat:

	name_pokemon character varying (50)	name_ability character varying (50)	slot integer	is_hidden boolean
1	bulbasaur	overgrow	1	false
2	bulbasaur	chlorophyll	3	true
3	ivysaur	overgrow	1	false
4	ivysaur	chlorophyll	3	true
5	venusaur	overgrow	1	false
6	venusaur	chlorophyll	3	true
7	charmander	blaze	1	false
8	charmander	solar power	3	true
9	charmeleon	blaze	1	false
10	charmeleon	solar power	3	true
11	charizard	blaze	1	false
12	charizard	solar power	3	true
13	squirtle	torrent	1	false
14	squirtle	rain dish	3	true
15	wartortle	torrent	1	false
16	wartortle	rain dish	3	true
17	blastoise	torrent	1	false
18	blastoise	rain dish	3	true
19	caterpie	shield dust	1	false
20	caterpie	run away	3	true
21	metapod	shed skin	1	false
22	butterfree	compound eyes	1	false
23	butterfree	tinted lens	3	true
24	weedle	shield dust	1	false

En la sisena consulta es vol buscar les relacions de mal que té un tipus seleccionat com a atacant contra qualsevol defensor. Es selecciona el nom del defensor i el multiplicador. S'agafa la taula TYPE_AGAINST i es posa com a atacant el que tingui el id que correpongui al nom del tipus donat

en el WHERE. Com a defensor es relaciona amb el id, però no es posa cap restricció per tal de que els mostri tots. Així és el resultat:

	defender character varying (50) 🔒	multiplier numeric 🔒
1	ground	2
2	rock	2
3	fire	2
4	dark	1
5	ice	1
6	psychic	1
7	electric	1
8	steel	1
9	ghost	1
10	bug	1
11	shadow	1
12	flying	1
13	fighting	1
14	normal	1
15	poison	1
16	unknown	1
17	fairy	1
18	water	0.5
19	grass	0.5
20	dragon	0.5

7.1.3 Explicació procés d'importació:

Entre les consultes realitzades es pot corroborar que la informació dels csv estan ben importades dins les taules ja que no falta cap dada a les taules, però també es pot comprovar que les diferents taules queden ben ellaçades, ja que entre totes les consultes veiem relacions entre taules, realitzades mitjançant JOIN. Algunes consultes es filtren per nom com per exemple del pokemon per mostrar que es pot buscar informació més específica de forma senzilla dins la nostra base de dades.

7.2 Entrenadors (1-2 pàgines)

7.2.1 Conjunt de consultes:

```
-- Aquesta comanda mostrara a l'entrenadora que hem escollit com a ratolí de laboratori per fer aquesta
-- comprovació. Hola Gema, benvinguda al mon POKEMON!
-- (C1)
SELECT * FROM TRAINER where id_trainer = 0;

-- Aquesta comanda mostrara als gym leaders importats al fitxer trainer.
-- (C2)
SELECT *
FROM TRAINER
where item_gift_leader is not null
;

-- Aquesta comanda mostrara que les id's dels gym leaders han estat ben relacionades amb la taula GYM
-- Podem veure com el nom coincideix amb la medalla que atorga en el joc, i tambe amb el nom del gimnas.
-- (C3)
SELECT GYM.id_gym, GYM.id_city, GYM.id_type, TRAINER.name_trainer, GYM.gym_name, GYM.gym_badge_leader
FROM GYM
JOIN TRAINER ON TRAINER.id_trainer = GYM.id_trainer
;
```

```

-- Aquesta comanda mostrara tots els entrenadors malvats, la seva id amb el seu nom i la id del seu company
-- i també el seu nom.
-- (C4)
SELECT EV1.id_trainer, TR1.name_trainer, EV1.id_partner, TR2.name_trainer
FROM EVIL_TRAINER as EV1
JOIN TRAINER as TR1 ON EV1.id_trainer = TR1.id_trainer
JOIN TRAINER as TR2 ON EV1.id_partner = TR2.id_trainer
;

-- Aquesta comanda mostrara la id dels entrenadors malvats i a quina organitzacio pertanyen així com el nom
-- d'aquesta i el del seu líder.
-- (C5)
SELECT EV.id_trainer, O.id_organization, O.leader_name, O.name_organization
FROM EVIL_TRAINER as EV
JOIN ORGANIZATION as O ON EV.id_organization = O.id_organization
;

-- Aquesta comanda mostrara tots els gimnasos que ha completat l'entrenador amb id (0..n). Podem veure que hi ha 32
-- columnes que coincideixen amb el nombre de gimnasos del fitxer gyms.csv
-- (C6)
SELECT * FROM BATTLES_IN WHERE id_trainer = 0;
-- També es interessant saber que al igual que en els jocs oficials, tant líders de gimnas com entrenadors
-- malvats no poden pas participar en gimnasos, i per tant no seran inclosos en aquesta taula.
-- per fer la demostracio probarem amb 2 id's de entrenadors amb aquestes caracteristiques i veurem
-- com no apareixen
-- (C7)
SELECT * FROM BATTLES_IN WHERE id_trainer = 403;
SELECT * FROM BATTLES_IN WHERE id_trainer = 500
;

-- Aquesta comanda mostrara quins pokemons ha capturat l'entrenador amb id (0..n). Per l'exemple posarem
-- a la nostra Gema!
-- (C8)
SELECT PC.id_pokemon_caught, PC.id_pokemon, PC.nick_name, PC.pokemon_status_condition, PC.pokemon_max_hp,
       PC.pokemon_remaining_hp
FROM POKEMON_CAUGHT as PC
JOIN TRAINER as TR ON TR.id_trainer = PC.id_trainer
WHERE TR.id_trainer = 0
ORDER BY 1
;

-- Veurem que aquells que tenen "null" en l'estatus seran els que no pertanyen a l'equip de Pokemon
-- que ara porta l'entrenadora amb ella. I els te guardats al PC del Centre Pokemon.
-- Amb la comanda que hi ha a continuacio posarem llum a la foscor, seleccionant els Pokemon que la
-- Gema porta ara mateix amb ella.
-- (C9)
SELECT TEAM.id_pokemon_caught, PC.nick_name, TEAM.slot
FROM TEAM
JOIN POKEMON_CAUGHT as PC ON TEAM.id_pokemon_caught = PC.id_pokemon_caught
JOIN TRAINER as TR ON TEAM.id_trainer = TR.id_trainer
WHERE TR.id_trainer = 0
ORDER BY 1
;

-- Ara anem a veure les estadistiques de combat de'n Shorty, Pokemon que pertany a l'equip de l'entrenadora
-- amb id 0, Gema! El pobre Shorty si ens fixem en comandes anteriors podem veure que ha estat congelat i
-- en aquests moments debilitat amb 0 PS restants.
-- Aquesta comanda ens mostrara que es el que ha passat.
-- (C10)
SELECT BE.id_battle, PC.nick_name, TR.name_trainer, BS.damage_received, BS.damage_dealt
FROM BATTLE_STATS as BS
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 POKEMON_CAUGHT as PC on BS.id_pokemon_caught = PC.id_pokemon_caught
WHERE TR.id_trainer = 0 and PC.nick_name LIKE '%Shorty%'
;

```



```

-- S'han disputat un total de 600 combats entre tots els entrenadors. La comanda que hi ha a
-- continuació provarà que no dic mentides.
-- (C11)
SELECT DISTINCT id_battle FROM BATTLE
ORDER BY 1 DESC
;

-- Per tant tindrem com a mínim informació de 600 combats en la taula de estadístiques, només
-- mostrarem 1 pokemon per combat independentment sigui de l'entrenador guanyador o perdedor
-- (C12)
SELECT DISTINCT ON (id_battle) id_pokemon_caught, id_battle FROM BATTLE_STATS
ORDER BY 2 DESC
;

-- i ara tots els pokemons de cada combat, al ser 6 Pokemon per 2 entrenadors tindrem 12 camps per
-- cada batalla realitzada (com a màxim). Seleccionarem el combat 600.
-- (C13)
SELECT id_pokemon_caught, id_battle FROM BATTLE_STATS
WHERE id_battle = 599
;

-- finalment, com tots els entrenadors tenen motxilla, independentment si són malevolts o líders de gimnas,
-- ja que tots poden tenir objectes en combat. Verificarem que tots els entrenadors, tingui objectes a la seva motxilla.
-- Aquesta comanda realitza un anàlisi exhaustiu de la bossa de tots els entrenadors (en el csv 8357 columnes)
-- (C14)
SELECT * FROM CONTAINS_ITEM as CI
JOIN BACKPACK as BK on BK.id_backpack = CI.id_backpack
JOIN TRAINER as TR on TR.id_trainer = BK.id_backpack
;

```

7.2.2 Peticions per separat:

Aquesta comanda mostrarà a l'entrenadora que hem escollit com a ratolí de laboratori per fer aquesta comprovació. Hola Gema benvinguda al món POKÉMON!

La idea d'aquesta comanda és merament informativa per poder comparar amb les comandes que hi ha a continuació i contrastar tota la informació proporcionada.

(C1)

SELECT * FROM TRAINER where id_trainer = 0;

	id_trainer [PK] integer	name_trainer character varying (50)	exp_trainer integer	pokecoins_trainer integer	class_trainer character varying (50)	phrase_trainer character varying (200)	item_gift_leader character varying (50)
1	0	Gema	536957	3820	Expert	[null]	[null]

Aquesta comanda mostrarà als “gym leaders” importats al fitxer “trainer”. Això ho fem perquè aquest ha estat el criteri per decidir quins entrenadors eren o no líders de gimnàs. D'aquesta manera ens assegurem que s'han triat bé. La idea darrera d'aquesta comanda és poder mostrar tots els líders de gimnàs i quedar-nos amb les seves ID's per poder contrastar informació (comanda C5).

(C2)

**SELECT * FROM TRAINER
WHERE item_gift_leader IS NOT null;**

	id_trainer [PK] integer	name_trainer character varying (50)	exp_trainer integer	pokecoins_trainer integer	class_trainer character varying (50)	phrase_trainer character varying (200)	item_gift_leader character varying (50)
1	500	Brock	133527	4304	Gym Leader	[null]	tm31
2	501	Misty	998037	10232	Gym Leader	[null]	tm17
3	502	Lt. Surge	1740508	7522	Gym Leader	[null]	tm47
4	503	Erika	1888802	3448	Gym Leader	[null]	tm23
5	504	Koga	1365804	5981	Gym Leader	[null]	tm49
6	505	Sabrina	1327375	2841	Gym Leader	[null]	tm11
7	506	Blaine	1884615	2450	Gym Leader	[null]	tm43
8	507	Giovanni	1692018	5185	Gym Leader and Rocket Leader	Hit the gym bro, escape the Matrix	hm05
9	508	Falkner	984460	3823	Gym Leader	[null]	hm02
10	509	Bugsy	1412378	9580	Gym Leader	[null]	tm38
11	510	Whitney	785912	6711	Gym Leader	[null]	hm05
12	511	Morty	1175748	11297	Gym Leader	[null]	tm55
13	512	Chuck	422541	8833	Gym Leader	[null]	hm01
14	513	Jasmine	779501	10997	Gym Leader	[null]	tm17
15	514	Pryce	571343	10103	Gym Leader	[null]	tm48
16	515	Clair	238082	7697	Gym Leader	[null]	tm25
17	516	Roxanne	1865031	6860	Gym Leader	[null]	tm74
18	517	Brawly	1881683	4006	Gym Leader	[null]	tm05
19	518	Wattson	1648938	8044	Gym Leader	[null]	tm22
Total rows: 32 of 32		Query complete 00:00:00.067					Ln 76,

Aquesta comanda mostrarà que les id's dels “gym leaders” han estat ben relacionades amb la taula GYM. Podem veure com el nom coincideix amb la medalla que atorga en el joc, i també amb el nom del gimnàs. La idea és poder fer el producte cartesià (join) de dues taules per poder mostrar que a la taula GYM apareguin correctament relacionats els entrenadors que tenen com a rol ser líders de gimnàs.

(C3)

```
SELECT GYM.id_gym, GYM.id_city, GYM.id_type, TRAINER.name_trainer,
       GYM.gym_name, GYM.gym_badge_leader
FROM GYM
JOIN TRAINER ON TRAINER.id_trainer = GYM.id_trainer;
```

	id_gym integer	id_city integer	id_type integer	name_trainer character varying (50)	gym_name character varying (50)	gym_badge_leader character varying (50)
1	1	1	3	Falkner	1st Johto League Gym	Zephyr Badge
2	2	5	11	Wallace	8th Hoenn League Gym	Rain Badge
3	3	7	12	Gardenia	2nd Sinnoh League Gym	Forest Badge
4	4	8	2	Brawly	2nd Hoenn League Gym	Knuckle Badge
5	5	9	14	Sabrina	6th Indigo League Gym	Marsh Badge
6	6	15	15	Candice	7th Sinnoh League Gym	Icicle Badge
7	7	16	13	Lt. Surge	3rd Indigo League Gym	Thunder Badge
8	8	17	6	Roxanne	1st Hoenn League Gym	Stone Badge
Total rows: 32 of 32		Query complete 00:00:00.055				

Aquesta comanda mostrarà tots els entrenadors malvats, la seva id amb el seu nom i la id del seu company i també el seu nom. Ho farem per comprovar que tenim entrenadors malvats i que aquests pertanyen a una organització (ho veureu a la C5). La idea és fer dos productes cartesians (joins) amb taules TRAINER per poder trobar el nom de l’entrenador malèvol i el del seu company.

(C4)

```
SELECT EV1.id_trainer, TR1.name_trainer, EV1.id_partner, TR2.name_trainer
FROM EVIL_TRAINER as EV1
```

```
JOIN TRAINER as TR1 ON EV1.id_trainer = TR1.id_trainer
JOIN TRAINER as TR2 ON EV1.id_partner = TR2.id_trainer;
```

	id_trainer integer	name_trainer character varying (50)	id_partner integer	name_trainer character varying (50)
1	400	Salvatore	406	Lucien
2	406	Lucien	400	Salvatore
3	401	Bennett	407	Brenton
4	407	Brenton	401	Bennett
5	402	Desire	403	Erin
6	403	Erin	402	Desire
7	404	Pennie	405	Lela
8	405	Lela	404	Pennie
Total rows: 100 of 100		Query complete 00:00:00.071		

Aquesta comanada mostrarà la id dels entrenadors malvats i a quina organització pertanyen així com el nom d'aquesta i el del seu líder. La comanda consisteix a enllaçar l'entrenador malèvol amb l'organització criminal, i mostrar a quina pertany. Això ho fem amb un producte cartesià (join).

(C5)

```
SELECT EV.id_trainer, O.id_organization, O.leader_name, O.name_organization
FROM EVIL_TRAINER as EV
JOIN ORGANIZATION as O ON EV.id_organization = O.id_organization;
```

	id_trainer integer	id_organization integer	leader_name character varying (50)	name_organization character varying (50)
1	406	7	Giovanni	Rocket
2	400	7	Giovanni	Rocket
3	407	9	Maxie	Magma
4	401	9	Maxie	Magma
5	403	10	Cyrus	Galactic
6	402	10	Cyrus	Galactic
7	405	8	Archie	Aqua
8	404	8	Archie	Aqua
Total rows: 100 of 100		Query complete 00:00:00.058		

Aquesta comanda mostrarà tots els gimnasos que ha completat l'entrenador amb id (0..n). Podem veure quan executem que hi ha 32 columnes, que coincideixen amb el nombre de gimnasos del fitxer gyms.csv. La idea consisteix a veure que en aquest cas hi ha el mateix nombre de taules com gimnasos hi ha per verificar que podrem guardar-nos si ha estat completat o no per l'entrenador/a.

(C6)

```
SELECT * FROM BATTLES_IN WHERE id_trainer = 0;
```

	id_gym [PK] integer	id_trainer [PK] integer	completed boolean
1	1	0	false
2	2	0	false
3	3	0	false
4	4	0	false
5	5	0	false
6	6	0	false
7	7	0	false
8	8	0	false
Total rows: 32 of 32		Query complete 00:00:00.063	

També és interessant saber que com també en els jocs oficials, tant líders de gimnàs com entrenadors malvats no poden pas participar en gimnasos, i per tant no seran inclosos en aquesta taula. Per fer la demostració provarem amb 2 id's d'entrenadors amb aquestes característiques i veurem com no apareixen.

(C7)

```
SELECT * FROM BATTLES_IN WHERE id_trainer = 403;
SELECT * FROM BATTLES_IN WHERE id_trainer = 500;
```

id_gym [PK] integer	id_trainer [PK] integer	completed boolean
------------------------	----------------------------	----------------------

Total rows: 0 of 0 Query complete 00:00:00.055

Aquesta comanda mostrarà quins Pokémons ha capturat l'entrenador amb id (0..n). Per l'exemple posarem a la nostra Gema! La idea d'aquesta comanda és demostrar que no tots els Pokémons que ha capturat l'entrenador estan en el seu equip en aquests moments. Si no que de la mateixa manera que en el joc, aquests romandran en el PC fins que es necessitin. Es fa un ORDER BY per mostrar de manera ordenada els Pokémons i poder-los comparar amb comandes posteriors amb més facilitat. (com ara la C9).

(C8)

```
SELECT PC.id_pokemon_caught, PC.id_pokemon, PC.nick_name,
       PC.pokemon_status_condition, PC.pokemon_max_hp,
       PC.pokemon_remaining_hp
```

```

FROM POKEMON_CAUGHT as PC
JOIN TRAINER as TR ON TR.id_trainer = PC.id_trainer
WHERE TR.id_trainer = 0
ORDER BY 1;

```

	id_pokemon_caught [PK] integer	id_pokemon integer	nick_name character varying (50)	pokemon_status_condition character varying (50)	pokemon_max_hp integer	pokemon_remaining_hp integer
1	0	36	Abigail	none	17	0
2	1	418	Shorty	freeze	8	0
3	2	319	Greta	none	0	0
4	3	343	Dee	none	0	0
5	4	189	Presley	none	8	0
6	5	215	Cassis	none	30	0
7	6	289	Lou	[null]	[null]	[null]
8	7	227	Jordan	[null]	[null]	[null]
Total rows: 10 of 10 Query complete 00:00:00.103						

Veurem que aquells que tenen "null" en l'estatus seran els que no pertanyen a l'equip de Pokémon que ara porta l'entrenadora amb ella. I els té guardats al PC del Centre Pokémon. Amb la comanda que hi ha a continuació posarem llum a la fosc, seleccionant els Pokémon que la Gema porta ara mateix amb ella. . Es fa un ORDER BY per mostrar de manera ordenada els Pokémon i poder-los comparar amb comandes anteriors amb més facilitat. (com ara la C8).
(C9)

```

SELECT TEAM.id_pokemon_caught, PC.nick_name, TEAM.slot
FROM TEAM
JOIN POKEMON_CAUGHT as PC ON TEAM.id_pokemon_caught =
PC.id_pokemon_caught
JOIN TRAINER as TR ON TEAM.id_trainer = TR.id_trainer
WHERE TR.id_trainer = 0
ORDER BY 1;

```

	id_pokemon_caught integer	nick_name character varying (50)	slot integer
1	0	Abigail	1
2	1	Shorty	2
3	2	Greta	3
4	3	Dee	4
5	4	Presley	5
6	5	Cassis	6
Total rows: 6 of 6 Query complete 00:00:00.109			

Ara anem a veure les estadístiques de combat d'en Shorty, Pokemon que pertany a l'equip de l'entrenadora amb id 0, Gema! El pobre Shorty, si ens fixem en comandes anteriors podem veure que ha estat congelat i en aquests moments debilitat amb 0 PS restants.

Aquesta comanda ens mostrarà què és el que ha passat. La idea en aquesta comanda és buscar de forma molt específica aquells registres de combats on el Pokémon amb sobrenom Shorty hagi participat.

(C10)

```

SELECT BE.id_battle, PC.nick_name, TR.name_trainer, BS.damage_received,
       BS.damage_dealt
FROM BATTLE_STATS as BS
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 POKEMON_CAUGHT as PC on BS.id_pokemon_caught = PC.id_pokemon_caught
WHERE TR.id_trainer = 0 and PC.nick_name LIKE '%Shorty%';

```

	id_battle integer	nick_name character varying (50)	name_trainer character varying (50)	damage_received integer	damage_dealt integer
1	147	Shorty	Gema	8	3
2	226	Shorty	Gema	8	8

S'han disputat un total de 600 combats entre tots els entrenadors. La comanda que hi ha a continuació provarà que no dic mentides. La idea és poder veure el nombre total de combats realitzats per poder contrastar la informació amb les següents comandes (com ara la C12 o C13). Es fa un ORDER BY DESC per poder fixar-nos en el fet que l'id va de 0 al 599.

(C11)

```

SELECT DISTINCT id_battle FROM BATTLE
ORDER BY 1 DESC
;

```

	id_battle integer
1	599
2	598
3	597
4	596
5	595
6	594
7	593
8	592

Total rows: 600 of 600

Per tant, tindrem com a mínim informació de 600 combats en la taula d'estadístiques, només mostrarem 1 Pokémon per combat independentment sigui de l'entrenador guanyador o perdedor. En aquesta comanda es busca corroborar que es guarda la informació de cada combat per Pokémon independent. (és una comanda molt similar a la C13).

(C12)

```

SELECT DISTINCT ON (id_battle) id_pokemon_caught, id_battle FROM BATTLE_STATS
ORDER BY 2 DESC;

```

	id_pokemon_caught [PK] integer	id_battle [PK] integer
1	7801	599
2	3346	598
3	3292	597
4	13388	596
5	12312	595
6	3172	594
7	848	593
8	9574	592
Total rows: 600 of 600		Query complete 00:00:00.060

I ara tots els Pokémons de cada combat, en ser 6 Pokémon per 2 entrenadors tindrem 12 camps per cada batalla realitzada (com a màxim). Seleccionarem el combat 600. La idea és mostrar tots els Pokémon que han disputat una batalla i no només quedar-nos en què hem de mostrar 1 Pokémon per batalla realitzada (és una comanda molt similar a la C12).

(C13)

```
SELECT id_pokemon_caught, id_battle FROM BATTLE_STATS
WHERE id_battle = 599
;
```

	id_pokemon_caught [PK] integer	id_battle [PK] integer
1	7801	599
2	7802	599
3	7803	599
4	7804	599
5	7805	599
6	7806	599
7	11075	599
8	11076	599
Total rows: 12 of 12		Query complete 00:00:00.073

Finalment, com tots els entrenadors tenen motxilla, independentment si són malèvol o líders de gimnàs, ja que tots poden tenir objectes en combat. Verificarem que tots els entrenadors, tingui objectes a la seva motxilla. Aquesta comanda fa una anàlisi exhaustiva de la bossa de tots els entrenadors (en el csv 8357 columnes).

(C14)

```
SELECT * FROM CONTAINS_ITEM as CI
JOIN BACKPACK as BK on BK.id_backpack = CI.id_backpack
JOIN TRAINER as TR on TR.id_trainer = BK.id_trainer;
```

	id_backpack integer	id_object integer	obtention_method character varying (50)	date_time date	id_backpack integer	id_trainer integer	id_trainer integer	name_trainer character varying (50)	exp_trainer integer	pokecoins_trainer integer
1	0	89	REWARD	2001-07-25	0	0	0	Gema	536957	3820
2	0	272	REWARD	2018-10-30	0	0	0	Gema	536957	3820
3	0	509	REWARD	2009-07-12	0	0	0	Gema	536957	3820
4	0	208	REWARD	2010-01-18	0	0	0	Gema	536957	3820
5	0	473	REWARD	2018-05-08	0	0	0	Gema	536957	3820
6	0	219	GIFTED	2020-06-03	0	0	0	Gema	536957	3820
7	0	442	REWARD	1997-06-05	0	0	0	Gema	536957	3820

Total rows: 1000 of 8357 Query complete 00:00:00.286 ✓ Successfully run. Total query runtime: 286 msec. 8357 rows affected. ✕ Ln 179, Col 1

7.2.3 Explicació procés d'importació:

Tal com hem vist el resultat ha estat satisfactori. Ara bé s'ha de tenir en compte que en els casos d'importacions quan tens relacions amb la comanda REFERENCE cap a una altra taula, l'ordre en què fas les importacions afecta i molt a l'hora d'agafar les dades. Doncs en el cas que no s'hagi importat primer el camp on es fa referència, molt possiblement, el producte cartesià que facis amb la taula no donarà resultats. Per tant, important seguir aquest ordre d'importació en el cas d'importat l'apartat de dades dels entrenadors. (les taules que no tenen comentaris l'ordre no és estrictament important, però sí que és millor si poden mantenir un ordre similar a aquest).

TRAINER, – primera sempre!

ORGANIZATION

EVIL_TRAINER

BATTLE

POKEMON_CAUGHT – contra abans millor!

TEAM,

BACKPACK,

CONTAINS_ITEM (necessita tenir ITEM abans),

BATTLES_IN,

BATTLE_STATS,

BUYS (necessita tenir ITEM abans)

7.3 Compres (1-2 pàgines)

7.3.1 Conjunt de consultes:

Per a comprovar que totes les dades al voltant de compres s'han emplenat correctament des dels CSV, hem realitzat les següents consultes:


```

-- QUERY 1 - ITEMS
SELECT name_object, quick_sell_price FROM ITEM
WHERE quick_sell_price is not NULL
ORDER BY ITEM.id_object ASC;

-- QUERY 2 - TREASURE
SELECT ITEM.name_object, collector_price FROM TREASURE
INNER JOIN ITEM ON TREASURE.id_treasure = item.id_object
ORDER BY ITEM.id_object asc;

--QUERY 3 - POKEBALL
SELECT ITEM.name_object, max_radium, min_radium FROM POKEBALL
INNER JOIN ITEM ON POKEBALL.id_pokeball = ITEM.id_object
ORDER BY POKEBALL.id_pokeball desc;

-- QUERY 4 - BERRY
SELECT name, growth_time, firmness, max_num_harvest FROM BERRY
INNER JOIN ITEM ON BERRY.id_berry = ITEM.id_object
WHERE ITEM.name_object = BERRY.name AND BERRY.firmness = 'soft' AND BERRY.max_num_harvest = 10
ORDER BY BERRY.id_berry ASC;

-- QUERY 5 - FLAVOUR
SELECT* FROM FLAVOUR
ORDER BY FLAVOUR.id_flavour ASC;

-- QUERY 6 - BERRY_FLAVOUR
SELECT BERRY.name, BERRY_FLAVOUR.boost, FLAVOUR.name FROM BERRY_FLAVOUR
INNER JOIN BERRY ON BERRY.id_berry = BERRY_FLAVOUR.id_berry
INNER JOIN FLAVOUR ON FLAVOUR.id_flavour = BERRY_FLAVOUR.id_flavour
WHERE BERRY.id_berry = BERRY_FLAVOUR.id_berry AND BERRY_FLAVOUR.boost < 15 AND FLAVOUR.name = 'spicy'
ORDER BY BERRY.name ASC;

-- QUERY 7 - HEAL
SELECT ITEM.name_object, HEAL.heal_points, HEAL.can_revive FROM HEAL
INNER JOIN ITEM ON HEAL.id_heal = ITEM.id_object
WHERE ITEM.id_object = HEAL.id_heal AND HEAL.heal_points IS NOT NULL
ORDER BY HEAL.id_heal ASC;

-- QUERY 8 - BOOSTER
SELECT BOOSTER.stat_increase_time, BOOSTER.id_booster,
BASE_STAT.name FROM BOOSTER
INNER JOIN ITEM ON BOOSTER.id_booster = ITEM.id_object
INNER JOIN BASE_STAT ON BASE_STAT.id_base_stat = BOOSTER.id_stat
WHERE ITEM.id_object = BOOSTER.id_booster
ORDER BY BOOSTER.id_booster ASC;

```

```

-- QUERY 9 - SELLS + STORE
SELECT ITEM.name_object, STORE.store_name, ITEM.base_price_object,
SELLS.discount, SELLS.stock FROM ITEM
INNER JOIN SELLS ON SELLS.id_object = ITEM.id_object
INNER JOIN STORE ON STORE.id_store = SELLS.id_store
WHERE ITEM.id_object = SELLS.id_object
ORDER BY ITEM.name_object ASC;

-- QUERY 11 - BUYS + TRAINER + ITEM + STORE
SELECT TRAINER.name_trainer, ITEM.name_object, ITEM.description_object,
ITEM.base_price_object, BUYS.amount, BUYS.discount,
BUYS.cost as TOTAL_PRICE, BUYS.date_time,
STORE.store_name FROM BUYS
INNER JOIN ITEM ON BUYS.id_object = ITEM.id_object
INNER JOIN TRAINER ON TRAINER.id_trainer = BUYS.id_trainer
INNER JOIN STORE ON STORE.id_store = BUYS.id_store
ORDER BY TRAINER.name_trainer ASC;

-- QUERY 10 - TMHM + MOVES
SELECT ITEM.name_object, POKEMON_MOVE.name_move
FROM TM_HM
INNER JOIN ITEM ON ITEM.id_object = TM_HM.id_mt
INNER JOIN POKEMON_MOVE ON POKEMON_MOVE.id_move = TM_HM.id_move
ORDER BY name_object asc;

-- GENERAL QUERIES
SELECT * FROM ITEM;
SELECT * FROM SELLS;
SELECT * FROM STORE;
SELECT * FROM BUYS;
SELECT * FROM BOOSTER;
SELECT * FROM TMHM;
SELECT * FROM TREASURE;
SELECT * FROM HEAL;
SELECT * FROM BERRY;
SELECT * FROM POKEBALL;
SELECT * FROM BERRY_FLAVOUR;
SELECT * FROM FLAVOUR;

```

7.3.2 Peticions per separat:

- **QUERY 1 – ITEMS**

Consulta:

```
-- QUERY 1 - ITEMS
SELECT name_object, quick_sell_price FROM ITEM
WHERE quick_sell_price is not NULL
ORDER BY ITEM.id_object ASC;
```

Resultat:

	name_object character varying (50)	quick_sell_price integer
1	tiny mushroom	1
2	big mushroom	1
3	pearl	1
4	big pearl	1
5	stardust	1
6	star piece	1
7	nugget	1
8	rare bone	1

Explicació:

Amb aquesta consulta es vol veure tots els items que existeixen. Es poden trobar buscant aquells en que el quick_sell_price és NULL. És per això que fem un select del nom de l'item i mirem que no sigui NULL l'atribut indicat. Ordenem els items pel seu ID.

- **QUERY 2 – TREASURE**

Consulta:

```
-- QUERY 2 - TREASURE
SELECT ITEM.name_object, collector_price FROM TREASURE
INNER JOIN ITEM on TREASURE.id_treasure = item.id_object
ORDER BY ITEM.id_object asc;
```

Resultat:

	name_object character varying (50)	collector_price integer
1	tiny mushroom	500
2	big mushroom	5000
3	pearl	1400
4	big pearl	7500
5	stardust	2000
6	star piece	9800
7	nugget	10000
8	rare bone	10000

Explicació:

Amb aquesta consulta es vol veure tots els treasures que hi ha en la bbdd. Per fer-ho, cal buscar dins la taula de *ITEMS* tots aquells que corresponen a un tresor. Els tresors s'obtenen buscant dins la taula dels objectes tots aquells que coincideixen amb els IDs de la taula de tresors.

• QUERY 3 – POKEBALL

Consulta:

```
--QUERY 3 - POKEBALL
SELECT ITEM.name_object, max_ratum, min_ratum FROM POKEBALL
INNER JOIN ITEM ON POKEBALL.id_pokeball = ITEM.id_object
ORDER BY POKEBALL.id_pokeball desc;
```

Resultat:

	name_object character varying (50)	max_ratum integer	min_ratum integer
1	sport ball	5	1
2	park ball	1	1
3	love ball	8	8
4	friend ball	1	200
5	fast ball	4	4
6	heavy ball	8	0
7	moon ball	4	4
8	level ball	8	2
9	lure ball	3	3
10	cherish ball	1	1
11	quick ball	4	1
12	heal ball	1	1
13	dusk ball	5	1
14	premier ball	1	1
15	luxury ball	1	1
16	timer ball	4	0
17	repeat ball	3	1
18	nest ball	9	1
19	dive ball	5	1
20	net ball	3	1
21	safari ball	5	1
22	poke ball	1	1
23	great ball	5	1
24	ultra ball	2	2
25	master ball	1	1

Explicació:

Volem seguir veient els diferents items que existeixen i en aquest cas, les pokeballs. Per a fer la consulta i comprovar que totes s'han importat correctament dins la seva taula. Les llistarem totes accedint a la taula *POKEBALL* i relacionant el id d'aquestes amb els diferents ids dels objectes. Quan ambdós ids coincideixen, voldrà dir doncs que és una pokeball. Fem un INNER JOIN per relacionar ambdues taules.

• QUERY 4 – BERRY

Consulta:

```
-- QUERY 4 - BERRY
SELECT name, growth_time, firmness, max_num_harvest FROM BERRY
INNER JOIN ITEM ON BERRY.id_berry = ITEM.id_object
WHERE ITEM.name_object = BERRY.name AND BERRY.firmness = 'soft' AND BERRY.max_num_harvest = 10
ORDER BY BERRY.id_berry ASC;
```

Resultat:

	name character varying (50)	growth_time integer	firmness character varying (50)	max_num_harvest integer
1	bluk berry	2	soft	10
2	rabuta berry	6	soft	10

Explicació:

Ara volem comprovar que s'han guardat les dades de la taula *BERRY* correctament. Per fer-ho, farem un JOIN on mirarem que el ID de la Berry coincideixi amb el ID de l'item. Com a la

taula *BERRY* hi ha molta informació, per a comprovar-ho només mostrarem uns camps concrets que són els que es veuen en la imatge de l'apartat resultat.

• QUERY 5 – FLAVOUR

Consulta:

```
-- QUERY 5 - FLAVOUR
SELECT* FROM FLAVOUR
ORDER BY FLAVOUR.id_flavour ASC;
```

Resultat:

	id_flavour [PK] integer	name character varying (50)
1	1	dry
2	2	sour
3	3	spicy
4	4	sweet
5	5	bitter

Explicació:

Volem veure tots els sabors que hi ha. Per fer-ho, directament hem de buscar tots els sabors a la taula fent un `SELECT *` (all).

• QUERY 6 – BERRY_FLAVOUR

Consulta:

```
-- QUERY 6 - BERRY_FLAVOUR
SELECT BERRY.name, BERRY_FLAVOUR.boost, FLAVOUR.name FROM BERRY_FLAVOUR
INNER JOIN BERRY ON BERRY.id_berry = BERRY_FLAVOUR.id_berry
INNER JOIN FLAVOUR ON FLAVOUR.id_flavour = BERRY_FLAVOUR.id_flavour
WHERE BERRY.id_berry = BERRY_FLAVOUR.id_berry AND BERRY_FLAVOUR.boost < 15 AND FLAVOUR.name = 'spicy'
ORDER BY BERRY.name ASC
LIMIT 10;
```

Resultat:

	name character varying (50)	boost integer	name character varying (50)
1	apicot berry	10	spicy
2	belue berry	10	spicy
3	charti berry	10	spicy
4	cheri berry	10	spicy
5	hondew berry	10	spicy
6	leppa berry	10	spicy
7	lum berry	10	spicy
8	nomel berry	10	spicy
9	oran berry	10	spicy
10	persim berry	10	spicy

Explicació:

El que ens interessa ara és mostrar les bayes i el sabor que tenen. Per fer-ho hem d'unir les taules *FLAVOUR* i *BERRY* fent dos `INNER JOIN`s. Per fer diferents verificacions, mostrarem només aquelles que els seu boost sigui `< 15` i que el sabor sigui 'spicy'. A més, com obtenim molts resultats, limitarem aquests a màxim 10.

• QUERY 7 – HEAL

Consulta:

```
-- QUERY 7 - HEAL
SELECT ITEM.name_object, HEAL.heal_points, HEAL.can_revive FROM HEAL
INNER JOIN ITEM ON HEAL.id_heal = ITEM.id_object
WHERE ITEM.id_object = HEAL.id_heal AND HEAL.heal_points IS NOT NULL
ORDER BY HEAL.id_heal ASC;
```

Resultat:

	name_object character varying (50)	heal_points integer	can_revive boolean
1	potion	20	false
2	hyper potion	200	false
3	super potion	50	false
4	fresh water	50	false
5	soda pop	60	false
6	lemonade	80	false
7	moomoo milk	100	false
8	energy powder	505510	false
9	energy root	200101015	false
10	revival herb	101015	true
11	berry juice	20	false

Explicació:

Per a comprovar que la taula *HEAL* ha desat les dades correctament, mostrarem el nom dels objectes de curació, la curació que otorguen i si permet reviure o no. Per fer-ho, caldrà unir les taules *ITEM* i *HEAL*.

• QUERY 8 – BOOSTER

Consulta:

```
-- QUERY 8 - BOOSTER
SELECT BOOSTER.stat_increase_time, BOOSTER.id_booster,
BASE_STAT.name FROM BOOSTER
INNER JOIN ITEM ON BOOSTER.id_booster = ITEM.id_object
INNER JOIN BASE_STAT ON BASE_STAT.id_base_stat = BOOSTER.id_stat
WHERE ITEM.id_object = BOOSTER.id_booster
ORDER BY BOOSTER.id_booster ASC;
```

Resultat:

	stat_increase_time integer	id_booster integer	name character varying (20)
1	6	57	attack
2	5	58	defense
3	31	59	speed
4	25	61	attack
5	23	62	defense

Explicació:

Com volem comprovar que la taula *BOOSTER* ha guardat les dades del csv correctament, el que farem serà, mitjançant un JOIN entre la taula *ITEM* i *BOOSTER*, mostrar diferent informació d'aquesta taula.

• QUERY 9 – SELLS + STORE

Consulta:

```
-- QUERY 9 - SELLS + STORE
SELECT ITEM.name_object, STORE.store_name, ITEM.base_price_object,
SELLS.discount, SELLS.stock FROM ITEM
INNER JOIN SELLS ON SELLS.id_object = ITEM.id_object
INNER JOIN STORE ON STORE.id_store = SELLS.id_store
WHERE ITEM.id_object = SELLS.id_object AND SELLS.discount > 50 AND SELLS.stock < 10
ORDER BY SELLS.discount ASC;
```

Resultat:

	name_object character varying (50)	store_name character varying (50)	base_price_object integer	discount integer	stock integer
1	elixir	Reilly-Kozey	3000	52	1
2	tm08	Oberbrunner-Hickle	50000	54	8
3	data card 03	Haley and Sons	0	58	4
4	tm36	Pfeffer Group	1000	59	4
5	wise glasses	Hills-Greenholt	4000	59	5

Explicació:

- **QUERY 10 – TMHM + MOVES**

Consulta:

```
-- QUERY 10 - TMHM + MOVES
SELECT ITEM.name_object, POKEMON_MOVE.name_move
FROM TM_HM
INNER JOIN ITEM ON ITEM.id_object = TM_HM.id_mt
INNER JOIN POKEMON_MOVE ON POKEMON_MOVE.id_move = TM_HM.id_move
ORDER BY name_object asc;
```

Resultat:

	name_object character varying (50)	name_move character varying (50)
1	hm01	cut
2	hm02	fly
3	hm03	surf
4	hm04	strength
5	hm05	flash
6	hm06	whirlpool
7	hm07	waterfall
8	hm08	dive
9	tm01	mega punch
10	tm02	razor wind
11	tm03	swords dance
12	tm04	whirlwind
13	tm05	mega kick
14	tm06	toxic
15	tm07	horn drill
16	tm08	body slam
17	tm09	take down
18	tm10	double edge
19	tm11	bubble beam
20	tm12	water gun
21	tm13	ice beam
22	tm14	blizzard
23	tm15	hyper beam

Explicació:

Aquesta consulta permet mostrar els objectes que ensenyen un moviment i el nom d'aquests. Per fer-ho, per una banda cal unir la taula *ITEM* amb la taula *POKEMON_MOVE* i la taula *TM_HM*. Per altra banda, cal cercar tots aquells objectes on el id de la taula *ITEM* i la taula *TM_HM* coincideixin i aquells on el moviment de la taula *TM_HM* coincideixi amb el moviment de la taula *POKEMON_MOVE*.

- **QUERY 11 – BUYS + TRAINER + ITEM + STORE**

Consulta:

```
-- QUERY 11 - BUYS + TRAINER + ITEM + STORE
SELECT TRAINER.name_trainer, ITEM.name_object, ITEM.description_object,
ITEM.base_price_object, BUYS.amount, BUYS.discount,
BUYS.cost as TOTAL_PRICE, BUYS.date_time,
STORE.store_name FROM BUYS
INNER JOIN ITEM ON BUYS.id_object = ITEM.id_object
INNER JOIN TRAINER ON TRAINER.id_trainer = BUYS.id_trainer
INNER JOIN STORE ON STORE.id_store = BUYS.id_store AND TRAINER.name_trainer = 'Adan'
ORDER BY TRAINER.name_trainer ASC;
```

Resultat:

	name_trainer character varying (50)	name_object character varying (50)	description_object character varying (1000)
1	Adan	awakening	Used on a party Pokémon: Cures sleep.
2	Adan	mosaic mail	Used to send short messages to other players via Pokémon trading. Trainer may compose a message from a finite list of wo
3	Adan	chople berry	Held in battle: When the holder would take super-effective fighting-type damage, it consumes this item to halve the amount c
4	Adan	deep sea scale	Held by Clamperl: Doubles the holder's Special Defense. Evolves the holder into Gorebyss when traded.
5	Adan	tm10	Teaches Hidden Power to a compatible Pokémon.
6	Adan	tm17	Teaches Protect to a compatible Pokémon.
7	Adan	tm53	Teaches Energy Ball to a compatible Pokémon.
8	Adan	tm60	Teaches Drain Punch to a compatible Pokémon.
9	Adan	tm82	Teaches Sleep Talk to a compatible Pokémon.
10	Adan	tm83	Teaches Natural Gift to a compatible Pokémon.
11	Adan	sport ball	Used by a trainer in battle: Attempts to catch a wild Pokémon. The wild Pokémon's catch rate is 1.5× normal.

base_price_object integer	amount integer	discount integer	total_price integer	date_time character varying (100)	store_name character varying (50)
200	1	0	2000	Thu Apr 11 02:53:39 CEST 2013	Monahan, Klein and Harvey
50	1	0	2000	Fri Aug 16 14:55:38 CEST 2013	Monahan, Klein and Harvey
80	10	0	2000	Wed Feb 05 06:43:48 CET 2020	Monahan, Klein and Harvey
2000	5	0	2000	Thu Jan 05 11:17:13 CET 2023	Monahan, Klein and Harvey
1000	10	21	1580	Tue Jun 18 07:43:16 CEST 2019	Monahan, Klein and Harvey
10000	1	0	2000	Thu Aug 04 07:11:33 CEST 2011	Monahan, Klein and Harvey
1000	1	18	1640	Sat May 13 08:06:17 CEST 2006	Monahan, Klein and Harvey
30000	2	44	1120	Sat Jul 25 10:06:42 CEST 2015	Monahan, Klein and Harvey
1000	10	0	2000	Sat Oct 09 09:52:15 CEST 1999	Monahan, Klein and Harvey
100000	2	60	800	Sat Aug 24 11:57:45 CEST 2002	Monahan, Klein and Harvey
300	2	0	2000	Sat Mar 12 04:22:03 CET 2022	Monahan, Klein and Harvey

Explicació:

Aquesta query era de les més complexes que hem fet ja que hem unit 4 taules alhora. Per fer-ho, s'han fet diferents joins, els quals es poden apreciar a la captura, i després hem mostrat per pantalla diferent informació com el nom de l'entrenador, l'objecte que compra, la descripció de l'item, el preu, la quantitat, el descompte obtingut, el preu final i la data quan va realitzar la compra. Com es volia mostrar tanta informació la qual es trobava en diferents taules, s'han hagut d'unir moltes entre elles.

7.3.3 Explicació procés d'importació

Després de realitzar les diferents queries, hem pogut comprovar que les dades s'han importat correctament. Aquest fet es pot afirmar ja que a mida que es feien consultes, s'anava comprovant els resultats obtinguts amb els diferents fitxers csv.

A part de totes les queries explicades anteriorment, també n'he fet algunes de generals en les que simplement s'ha fet el select d'una taula concreta al complet per a verificar de manera individual que totes les taules creades en aquest bloc, han sigut creades correctament.

Aquestes queries generals han sigut les següents:


```
-- GENERAL QUERIES
SELECT * FROM ITEM;
SELECT * FROM SELLS;
SELECT * FROM STORE;
SELECT * FROM BUYS;
SELECT * FROM BOOSTER;
SELECT * FROM TMHM;
SELECT * FROM TREASURE;
SELECT * FROM HEAL;
SELECT * FROM BERRY;
SELECT * FROM POKEBALL;
SELECT * FROM BERRY_FLAVOUR;
SELECT * FROM FLAVOUR;
```

7.4 Exploració (1-2 pàgines)

7.4.1 Conjunt de consultes:

```
-- QUERY 1 - REGION
(SELECT region FROM LOCATIONS_INSERTION)
EXCEPT
(SELECT region_name FROM REGION);
```

```
-- QUERY 2 - AREA
(SELECT region, area
FROM LOCATIONS_INSERTION)
EXCEPT
(SELECT region_name, area_name
FROM AREA
INNER JOIN REGION ON REGION.id_region = AREA.id_region);
```

```
-- QUERY 3 - SUBAREA
(SELECT
    region,
    area,
    subarea,
    subareaID
FROM
    LOCATIONS_INSERTION
WHERE
    subareaID IS NOT NULL)
EXCEPT
(SELECT
    region_name,
    area_name,
    subarea_name,
    id_subarea
FROM
    SUBAREA
    INNER JOIN AREA ON AREA.id_area = SUBAREA.id_area
    INNER JOIN REGION ON REGION.id_region = AREA.id_region);
```

```
-- QUERY 4 - CITY
(SELECT
    region,
    area,
    population
FROM
    LOCATIONS_INSERTION
WHERE
    population IS NOT NULL)
EXCEPT
(SELECT
    region_name,
    area_name,
    population
FROM
    CITY
    INNER JOIN AREA ON AREA.id_area = CITY.id_area
    INNER JOIN REGION ON REGION.id_region = AREA.id_region);
```

```
-- QUERY 5 - ROUTE
(SELECT
    LOWER(route) route,
    LOWER(north) north,
    LOWER(east) east,
    LOWER(west) west,
    LOWER(south) south,
    pavement
FROM
    ROUTE_INSERTION)
EXCEPT
(SELECT
    a1.area_name,
    north.area_name,
    east.area_name,
    west.area_name,
    south.area_name,
    pavement
FROM
    ROUTE
    INNER JOIN AREA AS a1 ON a1.id_area = ROUTE.id_area
    LEFT JOIN AREA AS north ON north.id_area = ROUTE.north
    LEFT JOIN AREA AS east ON east.id_area = ROUTE.east
    LEFT JOIN AREA AS west ON west.id_area = ROUTE.west
    LEFT JOIN AREA AS south ON south.id_area = ROUTE.south);
```

```

-- QUERY 6 - ENCOUNTERS
(SELECT
    pokemon,
    subareaID,
    method_to_find,
    condition_type,
    condition_value,
    chance, min_level,
    max_level
FROM
    ENCOUNTERS_INSERTION)
EXCEPT
(SELECT
    POKEMON.name_pokemon,
    ENCOUNTERS.id_subarea,
    method_to_find,
    condition_type,
    condition_value,
    chance,
    min_level,
    max_level
FROM
    ENCOUNTERS
    INNER JOIN POKEMON ON POKEMON.id_pokemon = ENCOUNTERS.id_pokemon
    INNER JOIN SUBAREA ON SUBAREA.id_subarea = ENCOUNTERS.id_subarea);

```

```

-- QUERY 7 - GYM
(SELECT
    leader,
    LOWER(location_name) location_name,
    LOWER(type_name) type_name,
    badge,
    gym_name
FROM
    GYM_INSERTION)
EXCEPT
(SELECT
    TRAINER.name_trainer,
    AREA.area_name,
    TYPE_POKEMON.name_type,
    gym_badge_leader,
    gym_name
FROM
    GYM
    INNER JOIN CITY ON CITY.id_city = GYM.id_city
    INNER JOIN AREA ON AREA.id_area = CITY.id_area
    INNER JOIN TYPE_POKEMON ON TYPE_POKEMON.id_type = GYM.id_type
    INNER JOIN TRAINER ON TRAINER.id_trainer = GYM.id_trainer);

```

```
-- GENERAL QUERIES
SELECT * FROM REGION;
SELECT * FROM AREA;
SELECT * FROM CITY;
SELECT * FROM GYM;
SELECT * FROM ROUTE;
SELECT * FROM SUBAREA;
SELECT * FROM ENCOUNTERS;
SELECT * FROM POKEMON;
SELECT * FROM TYPE_POKEMON;

SELECT * FROM ROUTE
INNER JOIN AREA ON AREA.id_area = ROUTE.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
WHERE AREA.area_name LIKE 'route 48';

SELECT * FROM ROUTE;

SELECT * FROM AREA WHERE area_name LIKE 'hoenn pokemon league' OR 'johto pokemon league';
```

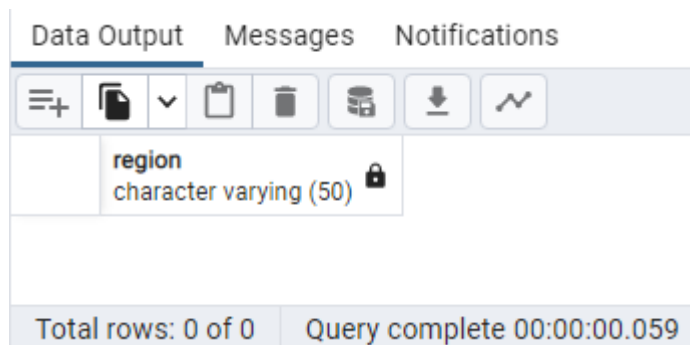
7.4.2 *Peticions per separat:*

- **QUERY 1 – REGION:**

1. Text de la comanda SQL:

```
(SELECT region FROM LOCATIONS_INSERTION)
EXCEPT
(SELECT region_name FROM REGION);
```

2. Captura del resultat de la seva execució:



3. Explicació de la petició:

L'idea de la petició es comparar la taula de copia (LOCATIONS_INSERTION), taula on es copia tot el contingut del CSV locations.csv, amb la taula REGION (aquesta taula conté un ID per cada regió, que es crea a l'hora d'inserir les dades, i el nom de la regió), fent que mostri totes les regions que no estiguin presents a la taula REGION. Si no es mostra cap atribut, llavors vol dir que la inserció de les dades es correcte.

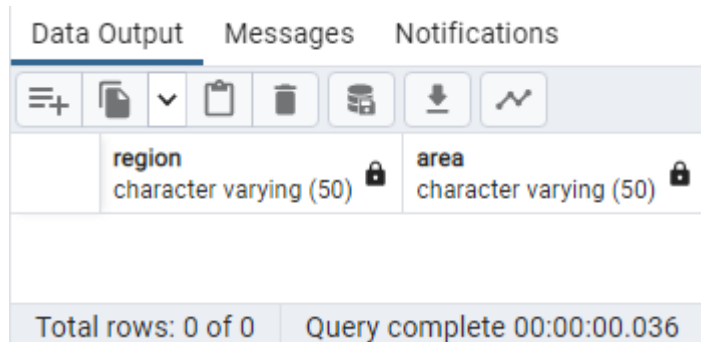
La petició fa un SELECT del nom de la regió a la taula de copia (LOCATIONS_INSERTION), després es fa un EXCEPT per excloure aquelles regions que també siguin presents a la taula REGION, fent un altre SELECT del nom de la regió, però aquesta vegada a la taula REGION.

- **QUERY 2 – AREA:**

1. Text de la comanda SQL:

```
(SELECT region, area
FROM LOCATIONS_INSERTION)
EXCEPT
(SELECT region_name, area_name
FROM AREA
INNER JOIN REGION ON REGION.id_region = AREA.id_region);
```

2. Captura del resultat de la seva execució:



region	area
character varying (50)	character varying (50)

Total rows: 0 of 0 Query complete 00:00:00.036

3. Explicació de la petició:

L'idea de la petició es comparar la taula de copia (LOCATIONS_INSERTION), taula on es copia tot el contingut del CSV locations.csv, amb la taula AREA (aquesta taula conté un ID per cada àrea, que es crea a l'hora d'inserir les dades, l'ID de la regió a la que es troba com a FK que s'importa de la taula REGION, i el nom de l'àrea), fent que mostri totes les regions i àrees que no estiguin presents a la taula REGION i AREA. Si no es mostra cap atribut, llavors vol dir que la inserció de les dades es correcte.

La petició fa un SELECT del nom de la regió i el nom de l'àrea a la taula de copia (LOCATIONS_INSERTION), després fa un EXCEPT per excloure aquelles regions i àrees que també siguin presents a les taules REGION i AREA, fent un altre SELECT del nom de la regió i de l'àrea, però aquesta vegada a la taula AREA.

Es fa un INNER JOIN amb la taula REGION per trobar a quina regió pertany l'àrea mitjançant els ID's de les taules REGION i AREA.

El motiu també utilitzar la taula REGION aquí es per validar que l'àrea té assignada la regió correcta.

- **QUERY 3 – SUBAREA:**

1. Text de la comanda SQL:

```
(SELECT
    region,
    area,
    subarea,
    subareaID
FROM
    LOCATIONS_INSERTION
```

```

WHERE
    subareaID IS NOT NULL)
EXCEPT
(SELECT
    region_name,
    area_name,
    subarea_name,
    id_subarea
FROM
    SUBAREA
    INNER JOIN AREA ON AREA.id_area = SUBAREA.id_area
    INNER JOIN REGION ON REGION.id_region = AREA.id_region);

```

2. Captura del resultat de la seva execució:

Data Output	Messages	Notifications
<div> <div> <div>+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>🗑️</div> <div>🔍</div> <div>⬇️</div> <div>📈</div> </div> </div>		
region	area	subarea
character varying (50)	character varying (50)	character varying (50)
		subareaid
		integer
Total rows: 0 of 0		
Query complete 00:00:00.035		

3. Explicació de la petició:

L'idea de la petició es comparar la taula de copia (LOCATIONS_INSERTION), taula on es copia tot el contingut del CSV locations.csv, amb la taula SUBAREA (aquesta taula conté un ID per cada subàrea, aquesta ID l'obtenim mitjançant la inserció de les dades de la taula de copia, l'ID de l'àrea a la que es troba com a FK que s'importa de la taula AREA, i el nom de la subàrea), fent que mostri totes les regions, àrees, subàrees i les ID's de les subàrees que no estiguin presents a la taula REGION, AREA i SUBÀREA. Si no es mostra cap atribut, llavors vol dir que la inserció de les dades es correcte.

La petició fa un SELECT del nom de la regió, el nom de l'àrea, el nom de la subàrea i l'ID de la subàrea a la taula de copia (LOCATIONS_INSERTION), després fa un EXCEPT per excloure aquelles regions, àrees, subàrees i ID's de les subàrees que també siguin presents a les taules REGION, AREA i SUBAREA, fent un altre SELECT del nom de la regió, de l'àrea, de la subàrea i de l'ID de la subàrea, però aquesta vegada a la taula SUBAREA. Es fa us de un WHERE i de dos INNER JOIN:

El WHERE s'utilitza per descartar tota definició d'àrea sense subàrea, ja que no te sentit guardar una subàrea que sigui NULL.

El primer un INNER JOIN amb la taula AREA per trobar a quina àrea pertany la subàrea mitjançant els ID's de les taules AREA i SUBAREA.

Després es fa un INNER JOIN amb la taula REGION per trobar a quina regió pertany l'àrea mitjançant els ID's de les taules REGION i AREA.

El motiu d'utilitzar la taula REGION i AREA aquí es per validar que la subàrea té assignada la regió i àrea correcta.

- **QUERY 4 – CITY:**

1. Text de la comanda SQL:

```
(SELECT
    region,
    area,
    population
FROM
    LOCATIONS_INSERTION
WHERE
    population IS NOT NULL)
EXCEPT
(SELECT
    region_name,
    area_name,
    population
FROM
    CITY
    INNER JOIN AREA ON AREA.id_area = CITY.id_area
    INNER JOIN REGION ON REGION.id_region = AREA.id_region);
```

2. Captura del resultat de la seva execució:

Data Output			Messages	Notifications
	region character varying (50) 🔒	area character varying (50) 🔒	population integer 🔒	
Total rows: 0 of 0 Query complete 00:00:00.054				

3. Explicació de la petició:

L'idea de la petició es comparar la taula de copia (LOCATIONS_INSERTION), taula on es copia tot el contingut del CSV locations.csv, amb la taula CITY (aquesta taula conté un ID per cada ciutat, que es crea a l'hora d'inserir les dades, l'ID de l'àrea que es una ciutat com a FK que s'importa de la taula AREA, i la població de la ciutat), fent que mostri totes les regions, àrees, i la població de cada ciutat que no estiguin presents a la taula REGION, AREA i CITY. Si no es mostra cap atribut, llavors vol dir que la inserció de les dades es correcte.

La petició fa un SELECT del nom de la regió, el nom de l'àrea (que es el nom de la ciutat), i la seva població a la taula de copia (LOCATIONS_INSERTION), després fa un EXCEPT per excloure aquelles regions, àrees (ciutats) i poblacions que també siguin presents a les taules REGION, AREA i CITY, fent un altre SELECT del nom

de la regió, de l'àrea (ciutat), i la població de la ciutat, però aquesta vegada a la taula CITY. Es fa us de un WHERE i dos INNER JOIN:

El WHERE s'utilitza per descartar totes aquelles àrees que no siguin ciutats, (ja que segons l'enunciat, es considera una ciutat tota aquella àrea que tingui una població, es a dir, que el camp de "population" del CSV locations.csv no sigui NULL), ja que aquestes no es guarden a la taula CITY perquè llavors no tindria cap mena de sentit aquesta taula.

El primer un INNER JOIN amb la taula AREA per trobar a quina àrea es una ciutat mitjançant els ID's de les taules AREA i CITY.

Després es fa un INNER JOIN amb la taula REGION per trobar a quina regió pertany l'àrea (ciutat) mitjançant els ID's de les taules REGION i AREA.

El motiu d'utilitzar la taula REGION i AREA aquí es per validar que la ciutat té assignada la regió i àrea correcta.

- **QUERY 5 – ROUTE:**

NOTA: EL CSV GYMS.CSV CONTÉ ÀREES INEXISTENTS AL CSV LOCATIONS.CSV, PER TANT ES MOSTRARÀN ERRORS AL RESULTAT DE LA COMPROBACIÓ. CONSIDERO QUE ELS CSV'S SON ERRONIS PERQUE A NINGUNA PART POSA QUE LES RUTES PODEN CONNECTAR A ÀREES INEXISTENTS A ALTRES CSV'S, PER TANT CONSIDERO QUE LES RUTES ES CONNECTEN A ÀREES CONEGUDES (QUE EXISTEIXIN EN EL CSV LOCATIONS). A L'EXPLICACIÓ DE LA IMPORTACIÓ S'EXPLICA TOT.

1. Text de la comanda SQL:

```
(SELECT
    LOWER(route) route,
    LOWER(north) north,
    LOWER(east) east,
    LOWER(west) west,
    LOWER(south) south,
    pavement
FROM
    ROUTE_INSERTION)
EXCEPT
(SELECT
    a1.area_name,
    north.area_name,
    east.area_name,
    west.area_name,
    south.area_name,
    pavement
FROM
    ROUTE
    INNER JOIN AREA AS a1 ON a1.id_area = ROUTE.id_area
    LEFT JOIN AREA AS north ON north.id_area = ROUTE.north
    LEFT JOIN AREA AS east ON east.id_area = ROUTE.east
```



```
LEFT JOIN AREA AS west ON west.id_area = ROUTE.west
LEFT JOIN AREA AS south ON south.id_area = ROUTE.south);
```

2. Captura del resultat de la seva execució:

Data Output Messages Notifications							
	route text	north text	east text	west text	south text	pavement character varying (50)	
1	route 28	[null]	hoenn pokemon league	mt silver	[null]	Grass	
2	route 22	[null]	viridian city	hoenn pokemon league	[null]	Gravel	
3	route 23	hoenn pokemon league	[null]	[null]	indigo plateau	Rocks	
4	route 48	johto safari zone	[null]	[null]	route 47	Gravel	
5	route 26	hoenn pokemon league	[null]	[null]	route 27	Grass	

Total rows: 5 of 5 Query complete 00:00:00.044

3. Explicació de la petició:

L'idea de la petició es comparar la taula de copia (ROUTE_INSERTION), taula on es copia tot el contingut del CSV routes.csv, amb la taula ROUTE (aquesta taula conté un ID per cada ruta, que es crea a l'hora d'inserir les dades, l'ID de l'àrea que es una ruta com a FK que s'importa de la taula AREA, quin àrea connecta per cada punt cardinal (nord, est, oest i sud), el tipus de paviment de la ruta i la seva distància (aquesta última no té un atribut específic al CSV routes.csv, llavors es genera amb la funció RANDOM()) en quilòmetres, per tant, no té sentit utilitzar aquest atribut per validar la inserció de les dades), fent que mostri totes les regions, àrees, connexions per cada punt cardinal i el tipus de paviment que no estiguin presents a la taula REGION, AREA i ROUTE. En aquest cas, com el csv gyms.csv conté un àrea que no apareix al csv locations.csv mostra que no existeixen certes rutes, si que existeixen, el que no existeixen son les àrees "hoenn pokemon league" i "johto safari zone", però el resultat es completament correcte, ja que no s'especifica a l'enunciat si les rutes poden connectar a àrees no definides.

La petició fa un SELECT del nom de la regió, el nom de l'àrea (que es el nom de la ruta), les seves connexions pels punts cardinals i el tipus de paviment a la taula de copia (ROUTE_INSERTION), després fa un EXCEPT per excloure aquelles regions, àrees (rutes), àrees connectades pels punts cardinals i el tipus de paviment que també siguin presents a les taules REGION, AREA i ROUTE, fent un altre SELECT del nom de la regió, de l'àrea (ruta), àrees connectades pels punts cardinals i el tipus de paviment, però aquesta vegada a la taula ROUTE. Es fa us de 4 INNER JOINS:

El primer INNER JOIN troba quines àrees de la taula AREA són rutes mitjançant els ID's de les taules AREA i ROUTE.

Els altres compleixen la missió de relacionar els ID's dels seus punts cardinals amb AREA.

El motiu d'utilitzar la taula REGION i AREA aquí es per validar que la ruta té assignada la regió i àrea correcta.

• **QUERY 6 – ENCOUNTERS**

1. Text de la comanda SQL:

```

(SELECT
    pokemon,
    subareaID,
    method_to_find,
    condition_type,
    condition_value,
    chance, min_level,
    max_level
FROM
    ENCOUNTERS_INSERTION)
EXCEPT
(SELECT
    POKEMON.name_pokemon,
    ENCOUNTERS.id_subarea,
    method_to_find,
    condition_type,
    condition_value,
    chance,
    min_level,
    max_level
FROM
    ENCOUNTERS
    INNER JOIN POKEMON ON POKEMON.id_pokemon =
ENCOUNTERS.id_pokemon
    INNER JOIN SUBAREA ON SUBAREA.id_subarea =
ENCOUNTERS.id_subarea);

```

2. Captura del resultat de la seva execució:

Data Output	Messages	Notifications
<div> <div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> </div> <div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> </div> </div>		
pokemon character varying (50)	subareaid integer	method_to_find character varying (50)
condition_type character varying (50)	condition_value character varying (50)	chance integer
min_level integer	max_level integer	
Total rows: 0 of 0 Query complete 00:00:00.044		

3. Explicació de la petició:

L'idea de la petició es comparar la taula de copia (ENCOUNTERS_INSERTION), taula on es copia tot el contingut del CSV encounters.csv, amb la taula ENCOUNTERS(aquesta taula conté l'ID de la subàrea on es pot donar la trobada com a FK que s'importa de la taula SUBAREA, l'ID del pokemon que es a la trobada com a FK que s'importa de la taula POKEMON, el mètode per trobar aquest pokemon, el tipus de condició per trobar-lo, el valor d'aquesta condició, la probabilitat de trobar-lo amb aquestes condicions, el nivell mínim i el nivell màxim amb el que es pot trobar), fent que mostri tots els pokemons, ID de les subàrees, mètodes per trobar, tipus de condicions, valor de les condicions, probabilitats, nivell mínim i nivell màxim que no estiguin presents a la taula POKEMON, SUBAREA i ENCOUNTERS. Si no es mostra cap atribut, llavors vol dir que la inserció de les dades es correcte.

La petició fa un SELECT del nom del pokemon, l'ID de la subàrea, el mètode per trobar-lo, el tipus de condició, el valor de la condició, la probabilitat, el nivell màxim i nivell mínim a la taula de copia (ENCOUNTERS_INSERTION) després fa un EXCEPT per excloure aquells pokemons, ID's de les subàrees, mètodes per trobar-los, tipus de condicions, valors de les condicions, les probabilitats, els nivells màxims i nivells mínims que també siguin presents a les taules POKEMON, SUBAREA i ENCOUNTERS, fent un altre SELECT del nom del pokemon, l'ID de la subàrea, el mètode per trobar-lo, el tipus de condició, el valor de la condició, la probabilitat, el nivell màxim i nivell mínim, però aquesta vegada a la taula ROUTE. Es fa us de dos INNER JOINS:

El primer INNER JOIN amb la taula POKEMON per trobar tots aquells pokemons que es troben a les trobades de cada subàrea mitjançant els ID's de les taules POKEMON i ENCOUNTERS.

El segon INNER JOIN amb la taula SUBAREA per trobar en quines subàrees es donen aquestes trobades mitjançant els ID's de les taules SUBAREA i ENCOUNTERS.

El motiu d'utilitzar la taula POKEMON i SUBAREA aquí es per validar que els pokemons i subàrees que s'estan guardant existeixin i siguin correctes.

- **QUERY 7 – GYM:**

1. Text de la comanda SQL:

```
(SELECT
    leader,
    LOWER(location_name) location_name,
    LOWER(type_name) type_name,
    badge,
    gym_name
FROM
    GYM_INSERTION)
EXCEPT
(SELECT
    TRAINER.name_trainer,
    AREA.area_name,
    TYPE_POKEMON.name_type,
    gym_badge_leader,
    gym_name
FROM
    GYM
    INNER JOIN CITY ON CITY.id_city = GYM.id_city
    INNER JOIN AREA ON AREA.id_area = CITY.id_area
    INNER JOIN TYPE_POKEMON ON TYPE_POKEMON.id_type = GYM.id_type
    INNER JOIN TRAINER ON TRAINER.id_trainer = GYM.id_trainer);
```

2. Captura del resultat de la seva execució:


Data Output

Messages


Notifications


☰


+





▼











<div>leader</div> <div>character varying (50)</div> <div>🔒</div>	<div>location_name</div> <div>text</div> <div>🔒</div>	<div>type_name</div> <div>text</div> <div>🔒</div>	<div>badge</div> <div>character varying (50)</div> <div>🔒</div>	<div>gym_name</div> <div>character varying (50)</div> <div>🔒</div>
--	---	---	---	--

Total rows: 0 of 0

Query complete 00:00:00.054

3. Explicació de la petició:

L'idea de la petició es comparar la taula de copia (GYM_INSERTION), taula on es copia tot el contingut del CSV gyms.csv, amb la taula GYM(aquesta taula conté l'ID del gimnàs, que es crea a l'hora d'inserir les dades, l'ID de la ciutat on es troba el gimnàs com a FK que s'importa de la taula CITY, l'ID del tipus de pokemon que utilitzen els rivals en el gimnàs com a FK que s'importa de la taula TYPE_POKEMON, l'ID de l'entrenador que es líder del gimnàs com a FK que s'importa de la taula TRAINER, el nom del gimnàs i la medalla que esta associada al gimnàs), fent que mostri tots els noms dels líders, ciutats, tipus de pokemon, medalles i noms de gimnàs que no estiguin presents a la taula GYM. Si no es mostra cap atribut, llavors vol dir que la inserció de les dades es correcte.

La petició fa un SELECT del nom del líder, la ciutat, el tipus de pokemon, la medalla i el nom del gimnàs a la taula de copia (GYM_INSERTION) després es fa un EXCEPT per excloure aquells líders, ciutats, tipus de pokemon, medalles, i noms de gimnàs que també estiguin presents a les taules CITY, TYPE_POKEMON, TRAINER i GYM, fent un altre SELECT del nom del líder, el tipus de pokemon, la medalla i el nom del gimnàs, però aquesta vegada a la taula GYM. Es ús de quatre INNER JOINS:

El primer INNER JOIN amb la taula CITY per trobar a quina ciutat es troba el gimnàs mitjançant els ID's de les taules CITY i GYM.

El segon INNER JOIN amb la taula AREA per trobar quin àrea es una ciutat mitjançant els ID's de les taules AREA i CITY.

El tercer INNER JOIN amb la taula TYPE_POKEMON per trobar quin es el tipus de pokemon que utilitzaran els rivals del gimnàs mitjançant els ID's de les taules TYPE_POKEMON i GYM.

I el quart INNER JOIN amb la taula TRAINER per trobar qui és el líder del gimnàs mitjançant els ID's de les taules TRAINER i GYM.

7.4.3 Explicació procés d'importació:

La meua importació ha sigut satisfactòria degut a que totes les proves amb les peticions anteriorment proporcionades no em donaven un resultat que no sigui satisfactori. He comprovat cada taula on he guardat les dades amb les taules que utilitzava per copiar les dades des de els CSV's i no m'han donat cap error. En cas de que no s'hagués importat qualsevol dada de la taula de copia a les taules finals, s'hagués mostrat amb les peticions anteriors.

També he comprovat amb peticions normals cada taula amb les següents peticions:

```
SELECT * FROM REGION;
SELECT * FROM AREA;
SELECT * FROM CITY;
SELECT * FROM GYM;
SELECT * FROM ROUTE;
SELECT * FROM SUBAREA;
SELECT * FROM ENCOUNTERS;
```

I he comprovat si les files corresponien amb el que hi havia en els CSV's, i les files corresponen tal i com han de ser amb els CSV's.

ERROR ALS CVS'S: El csv gyms.csv conté dues àrees que no existeixen al csv locations.csv, el que fa que no guardi aquelles connexions i segons la meua consulta per comprobar si s'ha importat correctament mostri que no s'han guardat certes coses.

Considero completament correcte que mostri que no ha guardat certes connexions de rutes degut a que considero que tota ruta ha de estar connectada a un àrea coneguda, per tant considero que el csv gyms.csv es incorrecte o el csv locations.csv es incorrecte. No s'aclareix en cap enunciat que les rutes poden estar connectado a àrees desconegudes. Adjunto les consultes per poder corroborar y comparar en els csv's que dues àrees no existeixen (**hoenn pokemon league i johto pokemon league**):

```
SELECT * FROM ROUTE
INNER JOIN AREA ON AREA.id_area = ROUTE.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
WHERE AREA.area_name LIKE 'route 48';
```

```
SELECT * FROM ROUTE;
```

```
SELECT * FROM AREA WHERE area_name LIKE 'hoenn pokemon league' OR 'johto pokemon league';
```

CAPTURES DELS CSV'S

gyms.csv

> hoenn pokemon league Aa .ab . * 1 of 4 ↑ ↓ ≡ ×

> johto safari zone Aa .ab . * 1 of 1 ↑ ↓ ≡ ×

locations.csv

> hoenn pokemon league Aa .ab . * No results ↑ ↓ ≡ ×

> johto safari zone Aa .ab . * No results ↑ ↓ ≡ ×

8 Conclusions

8.1 Recursos emprats

TODO: Introduir el número d'hores emprades per cada membre del grup en cada etapa de la fase 2 del projecte en una taula com aquesta:

Eta pa	Andrea	Pol	Joan	Leo	Total
Actualització dels models Entitat-Relació i Relacional	2.5h	3h	4h	1h	10.5h
Selecció del tipus de dades	0.5h	0.5h	0.5h	1h	2.5h
Codificació del model relacional	2.5h	2h	3h	2h	9.5h
Importació de la base de dades	29h	27h	28h	30h	107h
Validació de la base de dades	9h	8h	7h	15h	39h
Memòria	17h	17h	19h	20h	73h
Total:	60.5h	57.5h	59.5h	69h	241.5h

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.

8.2 Us d'IA (si cal, 1-2 pàgines)

No s'ha fet ús de cap tipus d'intel·ligència artificial, ni per crear les taules, ni per importar les dades, ni tampoc per validar les dades de la base de dades. Tota la informació de com crear els scripts ha estat preguntada als becaris i a classe

8.3 Lliçons apreses i conclusions (1 pàgina)

En aquesta pràctica hem après a base de creació de taules i importació de dades com d'important és tenir prèviament a la creació dels scripts, un model relacional i entitat-relació decent, per d'aquesta manera, amb petits canvis poder crear la base de dades sense grans complicacions.

Com teníem un model entitat-relació força correcte hem pogut arreglar el model relacional i adaptar-lo als requisits que se'ns demanava en els fitxers “.csv” d'aquesta pràctica. Amb unes “quèries” ben dissenyades hem pogut implementar i inserir totes les dades en la nostra base de dades PostgreSQL.

Hem après també que l'ordre és molt però que molt important a l'hora de la importació, doncs no és el mateix importar de llocs on ja hi ha informació que de llocs on no hi ha. Si s'està provant d'importar dades d'un lloc on no hi ha, no et quadraran cap de les comandes que facis amb productes cartesians doncs no et coincidiran de ninguna manera. Hem tingut molts errors debut l'ordre en el qual inseríem les coses i de debò que eren molt difícils de trobar perquè aparentment no fallava res, dit d'un altre manera, estava tota la comanda ben plantejada i estructurada. El que fallava es que ho comparàvem amb uns valors que encara no havien estat inserits a la taula en la que es feia referència.

En quant a coneixements adquirits, podem dir que gràcies a aquesta fase hem reforçat els coneixements relatius a crear taules, eliminar-les, inserir dades, alterar taules i fer consultes simples, que feia mesos que no practicàvem, perquè formava part del temari de principi de curs.

Creiem que tal com hem creat el model i hem generat el model físic, ens serà fàcil fer les consultes necessàries per aconseguir qualsevol informació concreta a alguna fila o taula de la base de dades.

Tenim ben definides les taules i relacions, cosa que permet una bona comunicació entre taules de la base de dades.

Finalment volíem aclarar que la majoria de canvis que hem hagut de fer al model, no es deuen tant a una mala planificació de la fase1, sinó que alguns requeriments de l'enunciat de la fase1 no concordaven amb les dades donades o a l'inrevés.