Function: get_pokemon_stats

Purpose:
This function retrieves the stats of a given Pokémon based on its ID.

Parameters:

- p_id (int): The ID of the Pokémon for which stats are to be retrieved.

Returns:

- stat_name (name): The name of the stat.
- stat_value (int): The value of the stat.

Example Usage:

```
SELECT * FROM get_pokemon_stats(1);
```

Description:

The function joins the **pokemon_stat** table with the **stat** table to fetch the names and values of the stats for a given Pokémon ID.

Function: get_pokemon_by_generation

Purpose:
This function retrieves all Pokémon that belong to a specific generation, including their names and other details.

Parameters:

- gen (int): The generation number.

Returns:

- pokemon_id (int): The ID of the Pokémon.
- pokemon_name (name): The name of the Pokémon.
- pokedex_info (text): Information from the Pokédex about the Pokémon.
- weight (numeric): The weight of the Pokémon.

- height (numeric): The height of the Pokémon.

Example Usage:

SELECT * FROM get_pokemon_by_generation(1);

Description:
The function joins the **pokemon** table with the **species** table to fetch the details of Pokémon from a specific generation, ensuring that the species name (i.e., the Pokémon name) is included in the results.

Function: get_effectiveness

Purpose:
This function retrieves the effectiveness multiplier of one type attacking another type.

Parameters:

- attacker (int): The ID of the attacking type.
- defender (int): The ID of the defending type.

Returns:

- multiplier (numeric): The effectiveness multiplier.

Example Usage:

SELECT get_effectiveness(1, 2);

Description:

The function queries the effectiveness table to find the multiplier for the effectiveness of one type attacking another. If no specific entry is found, it returns a default multiplier of 1.

Function: get_pokemon_abilities

Purpose:
This function retrieves all abilities of a given Pokémon, including whether they are hidden and the name of the Pokémon.

Parameters:

- p_id (int): The ID of the Pokémon.

Returns:

- pokemon_name (name): The name of the Pokémon.
- ability_name (name): The name of the ability.
- effect (text): The effect of the ability.
- is_hidden (boolean): Indicates if the ability is hidden.

Example Usage:

SELECT * FROM get_pokemon_abilities(1);

Description:

The function joins the **pokemon, species, pokemon_ability**, and **ability** tables to fetch the abilities of a given Pokémon, including the species name, ability names, their effects, and whether the abilities are hidden.

Function: get_pokemon_by_type

Purpose:
This function retrieves all Pokémon that belong to a specific type, including their names and other details.

Parameters:

- t_id (int): The ID of the type.

Returns:

- pokemon_id (int): The ID of the Pokémon.
- pokemon_name (name): The name of the Pokémon.
- pokedex_info (text): Information from the Pokédex about the Pokémon.
- weight (numeric): The weight of the Pokémon.
- height (numeric): The height of the Pokémon.

Example Usage:

SELECT * FROM get_pokemon_by_type(1);

Description:
The function joins the **pokemon**, **pokemon_type**, and **species** tables to fetch details of Pokémon of a specific type, ensuring that the species name is included in the results.

Trigger: check_weight_height

Purpose:
This trigger ensures that the weight and height of a Pokémon are positive values when a new Pokémon is inserted or an existing Pokémon is updated.

Trigger Timing:

- Before INSERT or UPDATE on the pokemon table.

Trigger Function:

- check_pokemon_weight_height

Example Usage:

This trigger is automatically invoked when inserting or updating the pokemon table.

Description:
This trigger function checks the weight and height fields of the pokemon table before an INSERT or UPDATE operation. If either the weight or height is less than or equal to zero, an exception is raised, preventing the operation from proceeding. This ensures that only valid positive values for weight and height are stored in the database.

Trigger: prevent_duplicate_type

Purpose:
This trigger ensures that a Pokémon cannot have duplicate entries for the same type in the pokemon_type table.

Trigger Timing:

- Before INSERT on the pokemon_type table.

Trigger Function:

- prevent_duplicate_pokemon_type

Example Usage:

This trigger is automatically invoked when inserting into the pokemon_type table.

Description:
This trigger function checks for existing entries in the pokemon_type table with the same pokemon_id and type_id as the new row being inserted. If a duplicate entry is found, it raises an exception, preventing the insertion. This ensures that each Pokémon can only have unique types, maintaining the integrity of the pokemon_type table.

Trigger: check_stat_value

Purpose:
This trigger ensures that the stat values for Pokémon are within a valid range (e.g., 0 to 255) whenever an insert or update operation is performed on the pokemon_stat table.

Trigger Timing:

- Before INSERT or UPDATE on the pokemon_stat table.

Trigger Function:

- validate_stat_value

Example Usage:

This trigger is automatically invoked when inserting or updating the pokemon_stat table.

Description:
This trigger function validate_stat_value checks whether the value being inserted or updated in the pokemon_stat table falls within the valid range of 0 to 255. If the value is outside this range, an exception is raised with an appropriate error message. This ensures that only valid stat values are allowed in the database, maintaining data integrity and consistency.

Procedure: type_change

Purpose:

The type_change procedure updates or inserts up to two types for a specified Pokémon.

Parameters:

- poke (int): The ID of the Pokémon whose types are to be updated.
- type1 (int): The ID of the primary type to be set for the Pokémon.
- type2 (int, default NULL): The ID of the secondary type to be set for the Pokémon. This parameter is optional.

Logic:

1. Update or Insert the First Type:
    - The procedure first checks if there is an existing type entry for the given Pokémon using a subquery with LIMIT 1.
    - If an entry exists, it updates the primary type.
    - If no entry exists, it inserts the primary type.
2. Update or Insert the Second Type (if provided):
    - The procedure checks if the type2 parameter is provided (type2 IS NOT NULL).
    - If there are more than one type entry for the Pokémon (COUNT(*) > 1), it updates the secondary type.
    - If no secondary type entry exists, it inserts the secondary type.

Procedure: update_pokemon_weight_and_height

Purpose:
The update_pokemon_weight_and_height procedure updates the weight and height of a specified Pokémon.

Parameters:

- poke_id (int): The ID of the Pokémon whose weight and height are to be updated.
- new_weight (numeric): The new weight of the Pokémon.
- new_height (numeric): The new height of the Pokémon.

Logic:

- The procedure updates the weight and height columns in the pokemon table for the Pokémon identified by poke_id.