Royal University of Bhutan

College of Science and Technology
Rinchending: Bhutan

DBS101
Database Systems Fundamentals
SS(2024)

Practical{4} Report

Submitted By;

Student Name : Tashi Penjor

Enrollment No.: 02230306

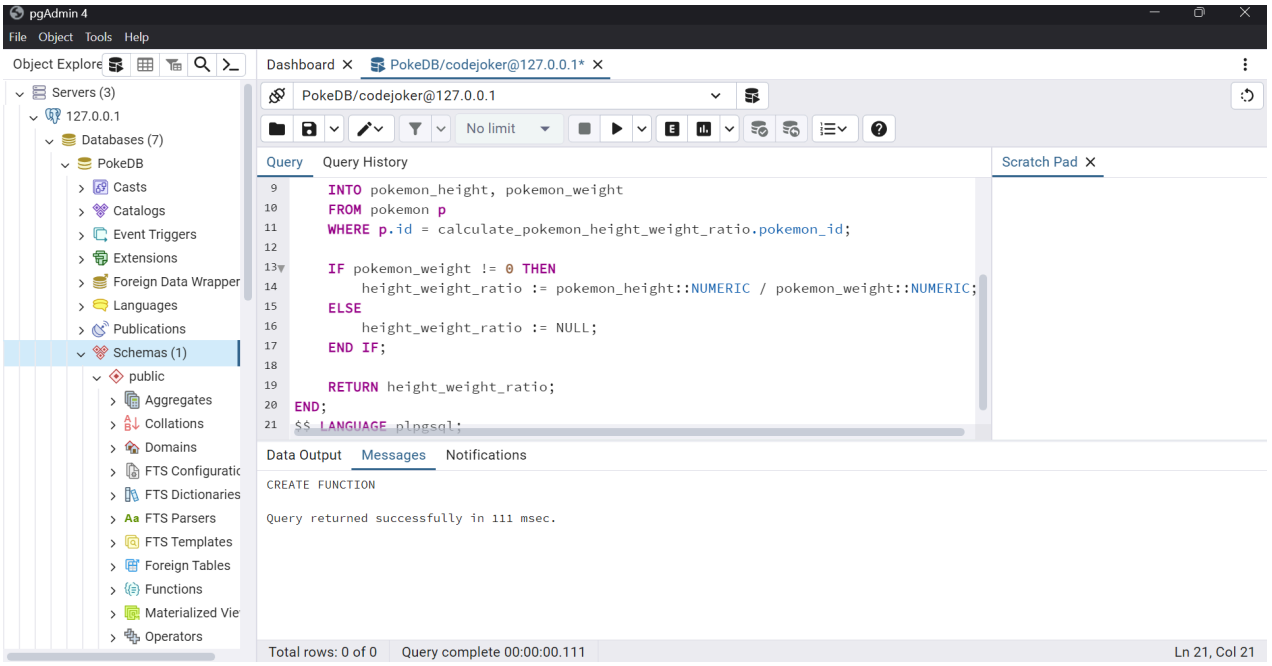
Programme : BESWE

Date : 21/03/2024

Table of content

Topic : Guided Session

Task 1 : The function calculates the height-weight ratio for the given Pokemon data by its ID. So the function extracts the height and weight of the Pokemon from the Pokemon CSV and then calculates the ratio by dividing the height by the weight. Moreover if the weight is zero it returns NULL.



The function also calls for the specific pokemon ID by using the SELECT function followed by the function name and ID.

On top of that the function calculates the height-weight ratio greater than 1.5 using the WHERE clause.

```
SELECT p.identifier, p.height, p.weight
FROM pokemon p
WHERE calculate_pokemon_height_weight_ratio(p.id) > 1.5;
```

Data Output Messages Notifications

	identifier character varying (100)	height integer	weight integer
1	gastly	13	1
2	haunter	16	1
3	cosmog	2	1
4	kartana	3	1
5	cursola	10	4
6	gastly	13	1
7	haunter	16	1
8	cosmog	2	1
9	kartana	3	1
10	cursola	10	4
11	gastly	13	1
12	haunter	16	1
13	cosmog	2	1
14	kartana	3	1
15	cursola	10	4

Task 2 : The function updates the base experience of a pokemon in the pokemon table based on the pokemon ID and the new experience from the given database.

```
CREATE OR REPLACE PROCEDURE update_pokemon_base_experience(
    pokemon_id INT,
    new_experience INT
)
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE pokemon p
    SET base_experience = update_pokemon_base_experience.new_experience
    WHERE p.id = update_pokemon_base_experience.pokemon_id;
END;
$$;
```

Data Output Messages Notifications

identifier character varying (100)	height integer	weight integer

Total rows: 0 of 0 Query complete 00:00:00.532

The function update the base experience of the pokemon with pokemon ID 1 to 200

```
CALL update_pokemon_base_experience(1, 200);
```

Data Output | Messages | Notifications

identifier character varying (100)	height integer	weight integer

Total rows: 0 of 0 | Query complete 00:00:00.504

The function updates the base experience of the pokemon and returns the updated pokemon record.

```
CREATE OR REPLACE FUNCTION update_and_get_pokemon(pokemon_id INT, new_experience IN
RETURNS pokemon
LANGUAGE plpgsql
AS $$
DECLARE
    updated_pokemon pokemon;
BEGIN
    CALL update_pokemon_base_experience(pokemon_id, new_experience);
    SELECT * INTO updated_pokemon FROM pokemon WHERE id = pokemon_id;
    RETURN updated_pokemon;
END;
$$;

SELECT * FROM update_and_get_pokemon(1, 250);
```

Data Output | Messages | Notifications

	id integer	identifier character varying (100)	species_id integer	height integer	weight integer	base_experience integer	order integer	is_default boolean
1	1	bulbasaur	1	7	69	250	1	true

Total rows: 1 of 1 | Query complete 00:00:00.455

Task 3 : The function returns the table of pokemon from the database filtered by a specific identifier. It includes the functions like pokemon ID, pokemon type, pokemon color and pokemon shape.

```
CREATE OR REPLACE FUNCTION get_pokemon_by_type(type_identifier VARCHAR(100))
RETURNS TABLE (
    pokemon_id INT,
    pokemon_identifier VARCHAR(100),
    pokemon_type VARCHAR(100),
    pokemon_color VARCHAR(100),
    pokemon_shape VARCHAR(100)
)
AS $$
BEGIN
    RETURN QUERY
    SELECT
        id,
        identifier,
        type
```

Data Output Messages Notifications

id integer	identifier character varying (100)	species_id integer	height integer	weight integer	base_experience integer	order integer	is_default boolean

Task 4 : The function prints the result up to a specific limit of the number of the pokemon in the pokemon table using the WHILE LOOP and FOR LOOP.

```
CREATE OR REPLACE FUNCTION print_numbers_up_to_limit(lmt INT)
RETURNS VOID AS $$
DECLARE
    counter INT := 1;
    pokemon_row RECORD;
BEGIN
    FOR pokemon_row IN SELECT id, identifier FROM pokemon LOOP
        WHILE counter <= lmt LOOP
            RAISE NOTICE 'Pokémon ID: %, Identifier: %, Number: %', pokemon_row.id,
            counter := counter + 1;
        END LOOP;
        counter := 1; -- Reset the counter for the next Pokémon
    END LOOP;
END;
$$ LANGUAGE plpgsql;
```

Data Output Messages Notifications

id integer	identifier character varying (100)	species_id integer	height integer	weight integer	base_experience integer	order integer	is_default boolean

The function prints the number of pokemon up to the specific limit for each pokemon in the database.

```
SELECT print_numbers_up_to_limit(5);
```

Data Output | Messages | Notifications

	print_numbers_up_to_limit void
1	

The function calculates the total base experience of all the pokemon in the database by summing up all the base experience and returns the total number of base experience.

```
CREATE OR REPLACE FUNCTION calculate_total_experience()
RETURNS INT AS $$
DECLARE
    total_experience INT := 0;
    pokemon_row pokemon;
BEGIN
    FOR pokemon_row IN SELECT * FROM pokemon LOOP
        total_experience := total_experience + pokemon_row.base_experience;
    END LOOP;
    RETURN total_experience;
END;
$$ LANGUAGE plpgsql;
```

Data Output | Messages | Notifications

print_numbers_up_to_limit void

```
SELECT calculate_total_experience();
```

Data Output | Messages | Notifications

	calculate_total_experience integer
1	524490

The function shows the pokemon color of pokemon based on the pokemon color ID in the database.

```
CREATE OR REPLACE FUNCTION get_pokemon_color_name(color_id INT)
RETURNS VARCHAR(100) AS $$
DECLARE
    color_name VARCHAR(100);
BEGIN
    SELECT identifier INTO color_name
    FROM pokemon_colors
```

Data Output | Messages | Notifications

calculate_total_experience integer

```
SELECT get_pokemon_color_name(1);
```

Data Output | Messages | Notifications

	get_pokemon_color_name character varying
1	black

Task 5 : The function calculates the number of pokemon from the database with a specific type. The function calls for the type identifier as grass from the pokemon type database.

```sql
CREATE OR REPLACE FUNCTION get_pokemon_count_by_type(type_identifier VARCHAR(100))
RETURNS INT AS $$
DECLARE
    pokemon_count INT := 0;
BEGIN
    SELECT COUNT(*) INTO pokemon_count
    FROM pokemon_with_type
    WHERE type = get_pokemon_count_by_type.type_identifier;
    RETURN pokemon_count;
END;
$$ LANGUAGE plpgsql;
```

Data Output | Messages | Notifications

	get_pokemon_color_name character varying

Data Output | Messages | Notifications

	get_pokemon_count_by_type integer
1	0

Task 6 : The function checks for the base experience of the pokemon, whether their base experience is negative or positive before updating or inserting into the database.

```sql
CREATE OR REPLACE FUNCTION update_pokemon_base_experience_trigger()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.base_experience < 0 THEN
        RAISE EXCEPTION 'Base experience cannot be negative.';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_base_experience
BEFORE INSERT OR UPDATE ON pokemon
FOR EACH ROW
EXECUTE FUNCTION update_pokemon_base_experience_trigger();
```

Data Output | Messages | Notifications

	get_pokemon_count_by_type integer
1	0

Task 7 : The function returns the queries that are written using the recursive function to show the result of a chain starting from the pokemon whose ID starts from 1 of a pokemon species from the database.

```sql
WITH RECURSIVE pokemon_evolution_chain AS (
    SELECT id, identifier, evolves_from_species_id
    FROM pokemon_species
    WHERE id = 1 -- Starting Pokemon ID
    UNION ALL
    SELECT ps.id, ps.identifier, ps.evolves_from_species_id
    FROM pokemon_species ps
    JOIN pokemon_evolution_chain pec ON ps.evolves_from_species_id = pec.id
)
SELECT *
FROM pokemon_evolution_chain;
```

Data Output Messages Notifications

id integer	identifier character varying (100)	evolves_from_species_id integer

```sql
SELECT *
FROM pokemon_evolution_chain;
WITH RECURSIVE pokemon_evolution_chain AS (
    SELECT id, identifier, evolves_from_species_id
    FROM pokemon_species
    WHERE id = 1
    UNION ALL
    SELECT ps.id, ps.identifier, ps.evolves_from_species_id
    FROM pokemon_species ps
    JOIN pokemon_evolution_chain pec ON ps.evolves_from_species_id = pec.id
)
SELECT *
FROM pokemon_evolution_chain;
```

Data Output Messages Notifications

id integer	identifier character varying (100)	evolves_from_species_id integer

Conclusion :

The guided session covers a wide range of tasks to manipulate and analyze using the SQL including calculating height-weight ratios , updating the base experience, filtering pokemon by identifiers and more. The task also includes looping to the queries to track the pokemon evolution chains.