



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



འབྲུག་རྒྱལ་འཛོམས་གཙུག་ལག་སློབ་མཁེ།

College of Science and Technology Rinchending: Bhutan

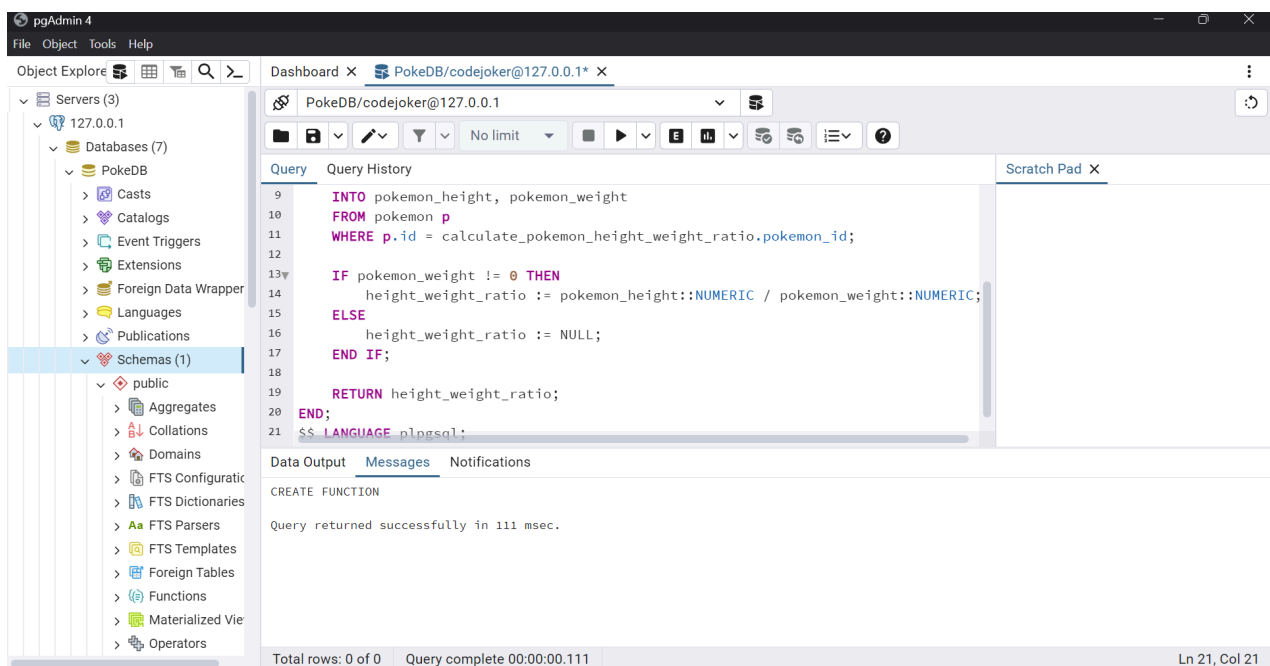


Table of content

SL.No	Topics	Page number
1	Guided session	2 - 9
2	Conclusion	10

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 screenshot shows the pgAdmin 4 interface. On the left, the 'Object Explorer' pane shows the database structure, including 'PokeDB' and 'Schemas (1)' with a 'public' schema. The main pane displays a SQL query for creating a function:

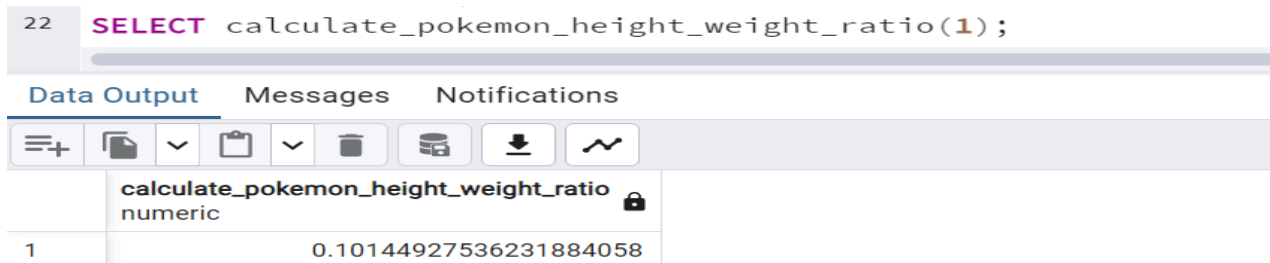
```

9  INTO pokemon_height, pokemon_weight
10 FROM pokemon p
11 WHERE p.id = calculate_pokemon_height_weight_ratio.pokemon_id;
12
13 IF pokemon_weight != 0 THEN
14     height_weight_ratio := pokemon_height::NUMERIC / pokemon_weight::NUMERIC;
15 ELSE
16     height_weight_ratio := NULL;
17 END IF;
18
19 RETURN height_weight_ratio;
20 END;
21 $$ LANGUAGE plpgsql;

```

Below the query editor, the 'Data Output' tab shows the message: 'Query returned successfully in 111 msec.'

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



The screenshot shows the 'Data Output' tab of pgAdmin 4. It displays the result of the query: `SELECT calculate_pokemon_height_weight_ratio(1);`. The result is a single row with the value 0.10144927536231884058.

	calculate_pokemon_height_weight_ratio numeric
1	0.10144927536231884058

College of Science and Technology

Rinchending: Bhutan

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

```
23 SELECT p.identifier, p.height, p.weight
24 FROM pokemon p
25 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.

```
26 CREATE OR REPLACE PROCEDURE update_pokemon_base_experience(
27     pokemon_id INT,
28     new_experience INT
29 )
30 LANGUAGE plpgsql
31 AS $$
32 BEGIN
33     UPDATE pokemon p
34     SET base_experience = update_pokemon_base_experience.new_experience
35     WHERE p.id = update_pokemon_base_experience.pokemon_id;
36 END;
37 $$;
```

Data Output Messages Notifications				
	identifier character varying (100)	height integer	weight integer	
Total rows: 0 of 0 Query complete 00:00:00.532				

College of Science and Technology

Rinchending: Bhutan

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

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

Data Output Messages Notifications

identifier	height	weight
character varying (100)	integer	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.

```
39 CREATE OR REPLACE FUNCTION update_and_get_pokemon(pokemon_id INT, new_experience INT)
40 RETURNS pokemon
41 LANGUAGE plpgsql
42 AS $$
43 DECLARE
44     updated_pokemon pokemon;
45 BEGIN
46     CALL update_pokemon_base_experience(pokemon_id, new_experience);
47     SELECT * INTO updated_pokemon FROM pokemon WHERE id = pokemon_id;
48     RETURN updated_pokemon;
49 END;
50 $$;
51
52 SELECT * FROM update_and_get_pokemon(1, 250);
```

Data Output Messages Notifications

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

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

College of Science and Technology

Rinchending: Bhutan

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.

```
53 CREATE OR REPLACE FUNCTION get_pokemon_by_type(type_identifier VARCHAR(100))
54 RETURNS TABLE (
55     pokemon_id INT,
56     pokemon_identifier VARCHAR(100),
57     pokemon_type VARCHAR(100),
58     pokemon_color VARCHAR(100),
59     pokemon_shape VARCHAR(100)
60 )
61 AS $$
62 BEGIN
63     RETURN QUERY
64     SELECT
65         id,
66         identifier,
67         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.

```
76 CREATE OR REPLACE FUNCTION print_numbers_up_to_limit(lmt INT)
77 RETURNS VOID AS $$
78 DECLARE
79     counter INT := 1;
80     pokemon_row RECORD;
81 BEGIN
82     FOR pokemon_row IN SELECT id, identifier FROM pokemon LOOP
83         WHILE counter <= lmt LOOP
84             RAISE NOTICE 'Pokémon ID: %, Identifier: %, Number: %', pokemon_row.id,
85                 counter := counter + 1;
86         END LOOP;
87         counter := 1; -- Reset the counter for the next Pokémon
88     END LOOP;
89 END;
```

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

College of Science and Technology

Rinchending: Bhutan

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

```
91 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.

```
92 CREATE OR REPLACE FUNCTION calculate_total_experience()
93 RETURNS INT AS $$
94 DECLARE
95     total_experience INT := 0;
96     pokemon_row pokemon;
97 BEGIN
98     FOR pokemon_row IN SELECT * FROM pokemon LOOP
99         total_experience := total_experience + pokemon_row.base_experience;
100     END LOOP;
101     RETURN total_experience;
102 END;
103 $$ LANGUAGE plpgsql;
```

Data Output		Messages	Notifications
	print_numbers_up_to_limit void		

```
104 SELECT calculate_total_experience();
```

Data Output		Messages	Notifications
	calculate_total_experience integer		
1	524490		

College of Science and Technology

Rinchending: Bhutan

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

```

105 CREATE OR REPLACE FUNCTION get_pokemon_color_name(color_id INT)
106 RETURNS VARCHAR(100) AS $$
107 DECLARE
108     color_name VARCHAR(100);
109 BEGIN
110     SELECT identifier INTO color_name
111     FROM pokemon_colors

```

Data Output Messages Notifications



calculate_total_experience
integer

```

121 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.

College of Science and Technology

Rinchending: Bhutan

```

122 CREATE OR REPLACE FUNCTION get_pokemon_count_by_type(type_identifier VARCHAR(100))
123 RETURNS INT AS $$
124 DECLARE
125     pokemon_count INT := 0;
126 BEGIN
127     SELECT COUNT(*) INTO pokemon_count
128     FROM pokemon_with_type
129     WHERE type = get_pokemon_count_by_type.type_identifier;
130     RETURN pokemon_count;
131 END;
132 $$ 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.

```

2 CREATE OR REPLACE FUNCTION update_pokemon_base_experience_trigger()
3 RETURNS TRIGGER AS $$
4 BEGIN
5     IF NEW.base_experience < 0 THEN
6         RAISE EXCEPTION 'Base experience cannot be negative.';
7     END IF;
8     RETURN NEW;
9 END;
10 $$ LANGUAGE plpgsql;
11
12 CREATE TRIGGER check_base_experience
13 BEFORE INSERT OR UPDATE ON pokemon
14 FOR EACH ROW
15 EXECUTE FUNCTION update_pokemon_base_experience_trigger();

```

Data Output Messages Notifications



get_pokemon_count_by_type
integer

1	0
---	---

College of Science and Technology

Rinchending: Bhutan

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.

```

1  WITH RECURSIVE pokemon_evolution_chain AS (
2      SELECT id, identifier, evolves_from_species_id
3      FROM pokemon_species
4      WHERE id = 1 -- Starting Pokemon ID
5      UNION ALL
6      SELECT ps.id, ps.identifier, ps.evolves_from_species_id
7      FROM pokemon_species ps
8      JOIN pokemon_evolution_chain pec ON ps.evolves_from_species_id = pec.id
9  )
10 SELECT *
11 FROM pokemon_evolution_chain;

```

Data Output Messages Notifications



	id	identifier	evolves_from_species_id
	integer	character varying (100)	integer

```

10 SELECT *
11 FROM pokemon_evolution_chain;
12 WITH RECURSIVE pokemon_evolution_chain AS (
13     SELECT id, identifier, evolves_from_species_id
14     FROM pokemon_species
15     WHERE id = 1
16     UNION ALL
17     SELECT ps.id, ps.identifier, ps.evolves_from_species_id
18     FROM pokemon_species ps
19     JOIN pokemon_evolution_chain pec ON ps.evolves_from_species_id = pec.id
20 )
21 SELECT *
22 FROM pokemon_evolution_chain;

```

Data Output Messages Notifications



	id	identifier	evolves_from_species_id
	integer	character varying (100)	integer



College of Science and Technology Rinchending: Bhutan

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.