Royal University of Bhutan

འབྲུག་རྒྱལ་འཛིན་གཙུག་ལག་སློབ་སྡེ།།

College of Science and Technology
Rinchending: Bhutan

DBS101

Database Systems Fundamentals

SS(2024)

Practical{5}   Report

Submitted By;

Student Name : Tashi Penjor

Enrollment No.: 02230306

Programme : BESWE

Date : 25/03/2024

Royal University of Bhutan

འབྲུག་རྒྱལ་འཛིན་གཙུག་ལག་སློབ་སྡེ།།

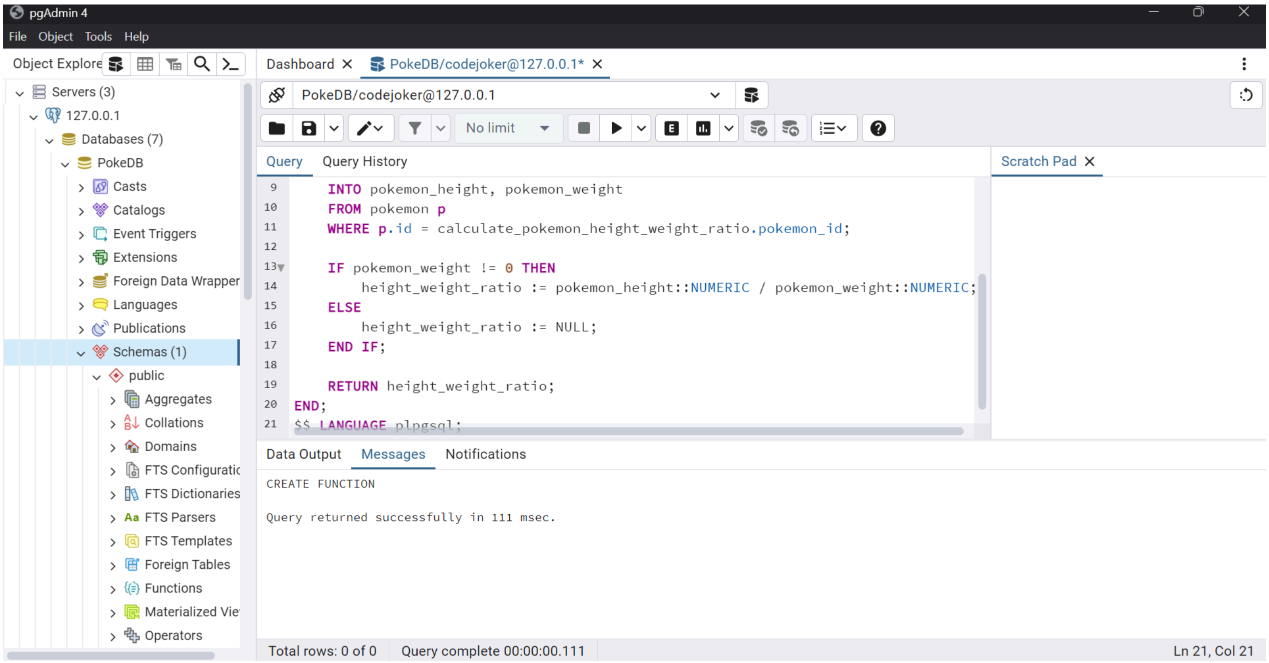
College of Science and Technology

Rinchending: Bhutan

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.

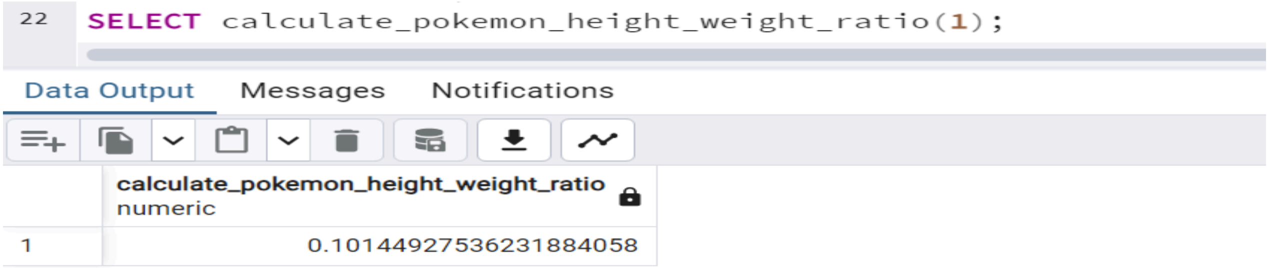


```
    INTO pokemon_height, pokemon_weight
    FROM pokemon p
    WHERE p.id = calculate_pokemon_height_weight_ratio.pokemon_id;

    IF pokemon_weight != 0 THEN
        height_weight_ratio := pokemon_height::NUMERIC / pokemon_weight::NUMERIC;
    ELSE
        height_weight_ratio := NULL;
    END IF;

    RETURN height_weight_ratio;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE FUNCTION

Query returned successfully in 111 msec.
```

Total rows: 0 of 0   Query complete 00:00:00.111   Ln 21, Col 21

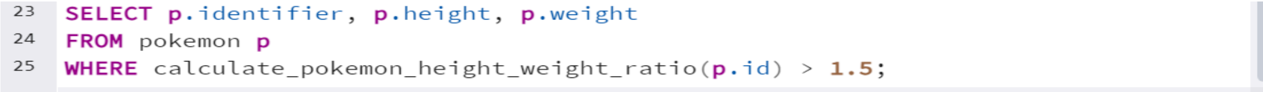
The function also calls for the specific pokemon ID by using the SELECT function followed by the function name and ID.



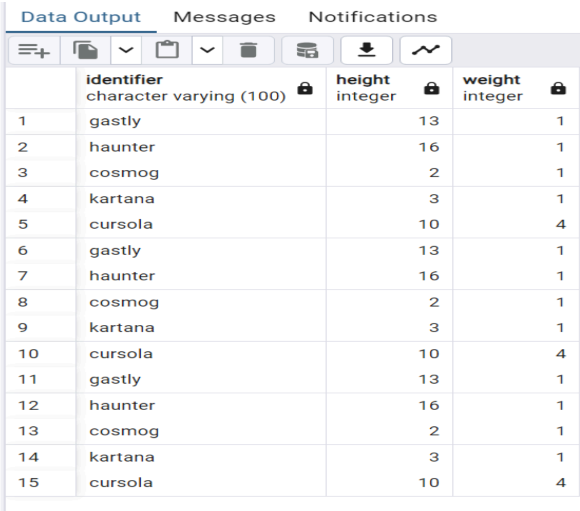
```
SELECT calculate_pokemon_height_weight_ratio(1);
```

	calculate_pokemon_height_weight_ratio numeric
1	0.10144927536231884058

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

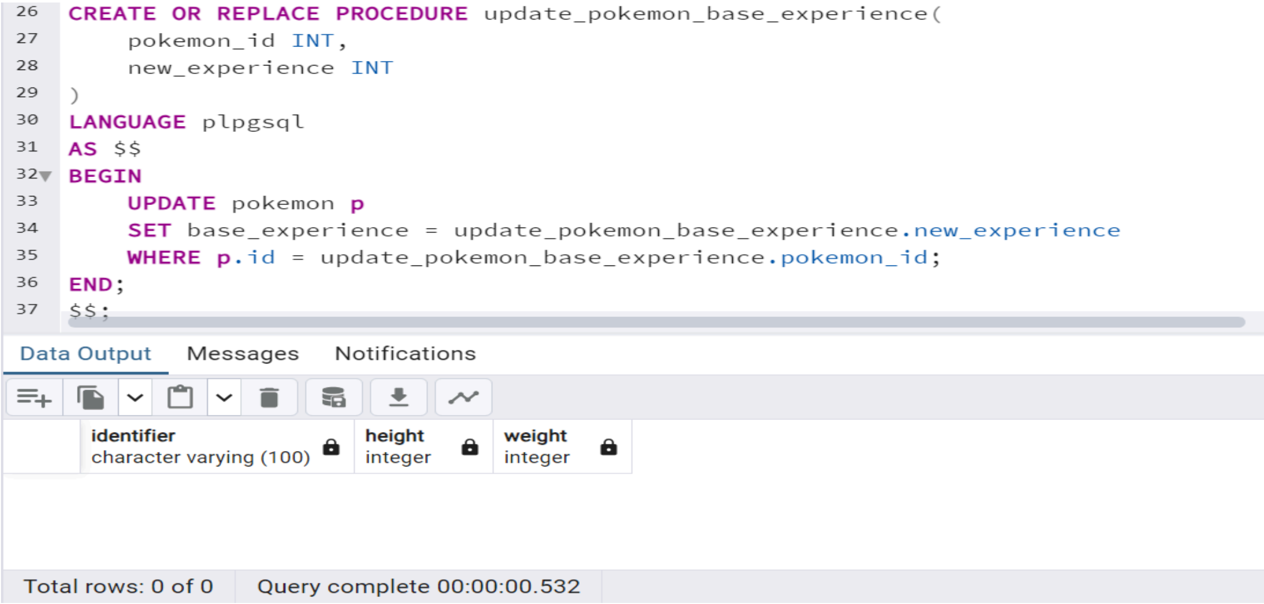


```sql
SELECT p.identifier, p.height, p.weight
FROM pokemon p
WHERE calculate_pokemon_height_weight_ratio(p.id) > 1.5;
```



```
     identifier                  height     weight
     character varying (100)     integer    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
```

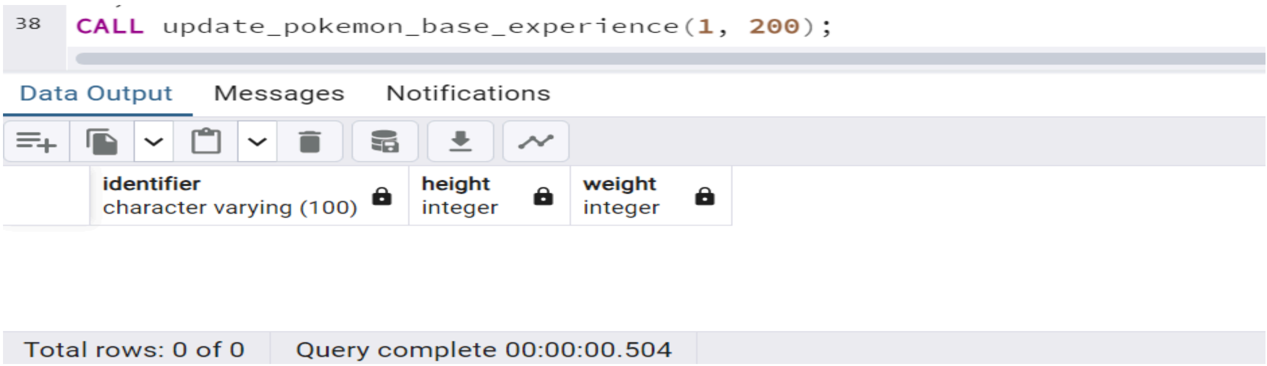
Task 2 : The function updates the base experience of a pokemon in the pokemon table based on the pokemon ID and the new experience from the given database.



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

```
     identifier                  height     weight
     character varying (100)     integer    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

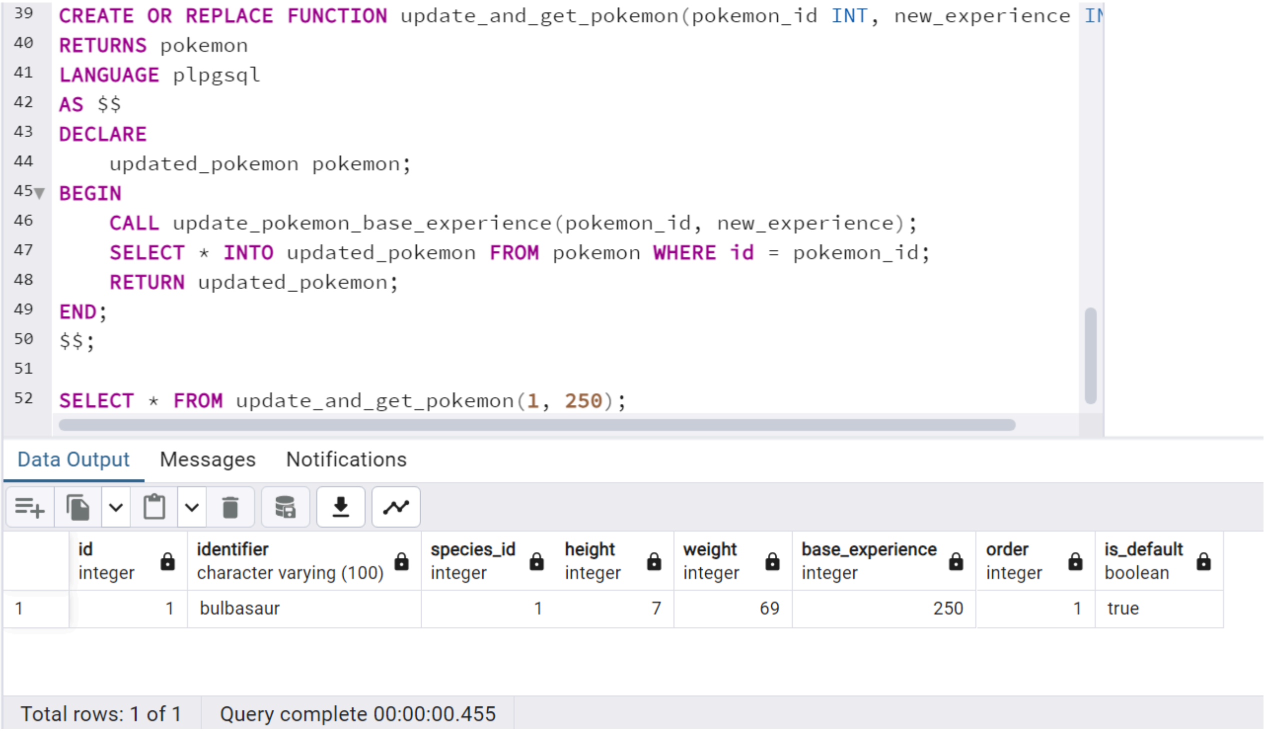


```
CALL update_pokemon_base_experience(1, 200);
```

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.

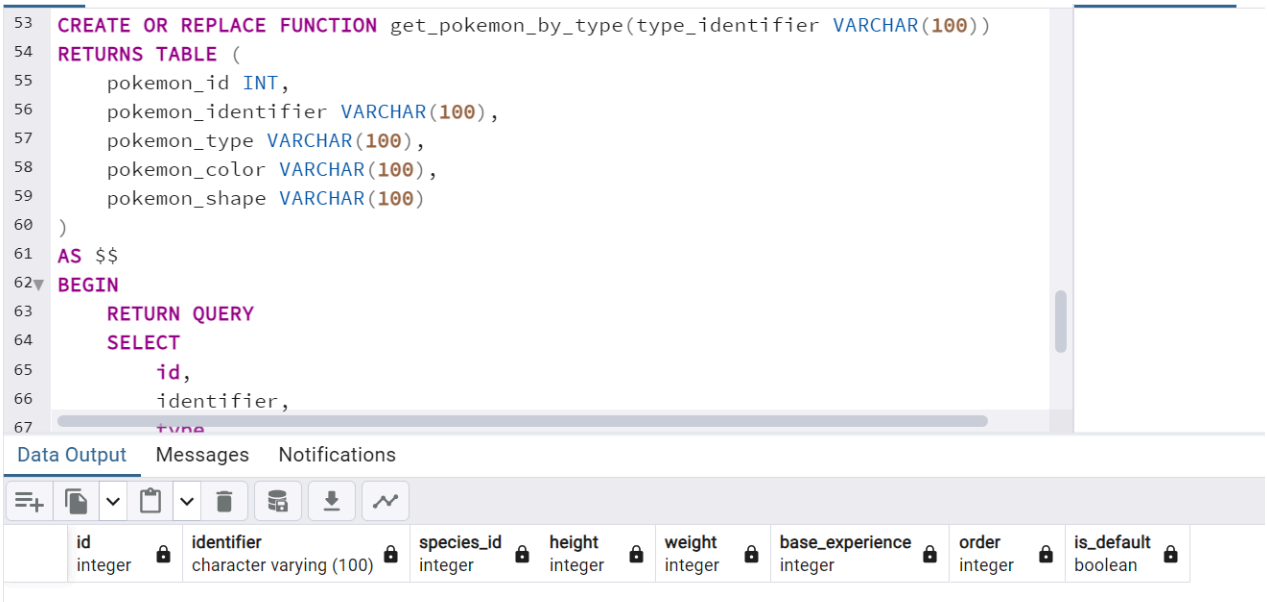


```
CREATE OR REPLACE FUNCTION update_and_get_pokemon(pokemon_id INT, new_experience IN
RETURNS pokemon
LANGUAGE plpgsql
AS $$
DECLARE
    updated_pokemon pokemon;
BEGIN
    CALL update_pokemon_base_experience(pokemon_id, new_experience);
    SELECT * INTO updated_pokemon FROM pokemon WHERE id = pokemon_id;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

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

	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

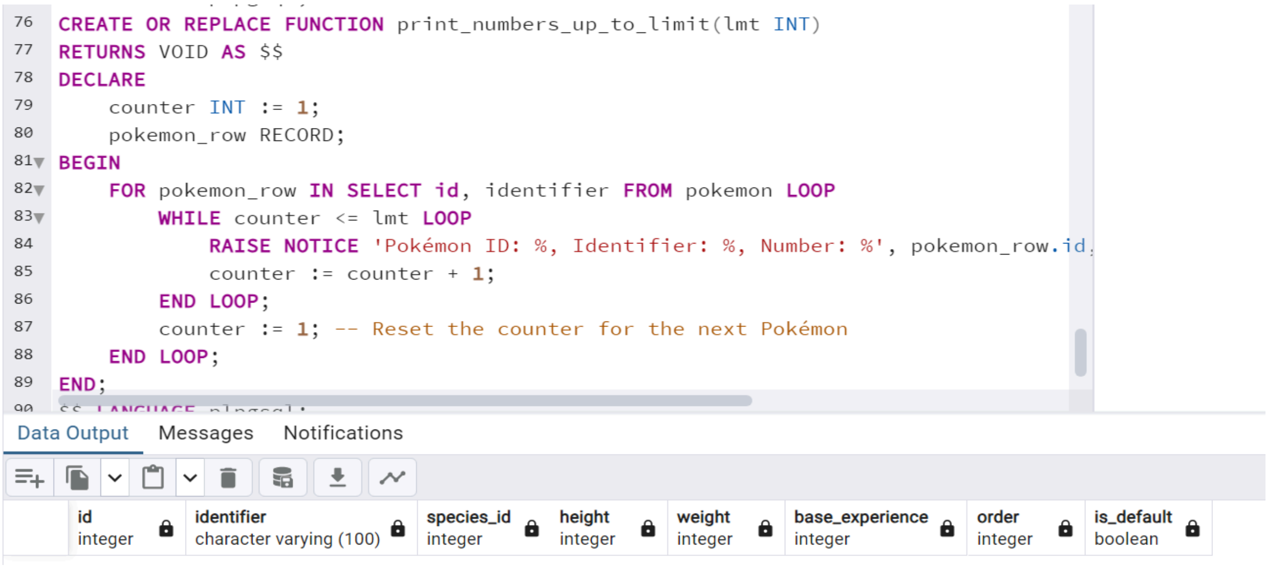
Task 3 : The function returns the table of pokemon from the database filtered by a specific identifier. It includes the functions like pokemon ID, pokemon type, pokemon color and pokemon shape.



```sql
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
```

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

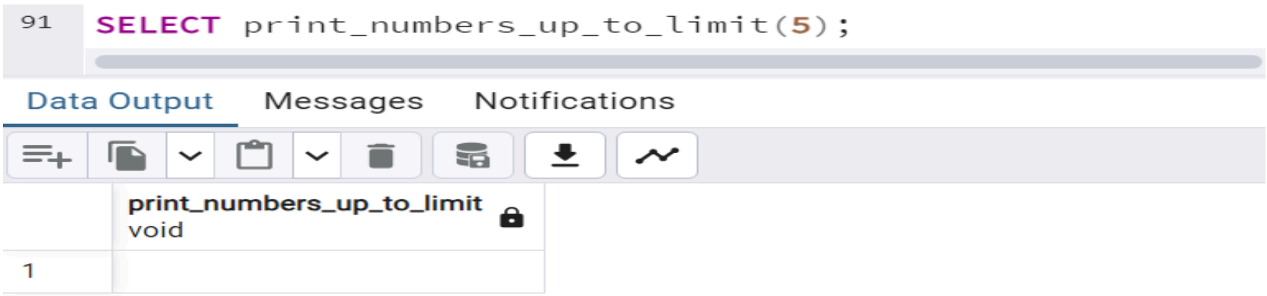
Task 4 : The function prints the result up to a specific limit of the number of the pokemon in the pokemon table using the WHILE LOOP and FOR LOOP.



```sql
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;
```

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

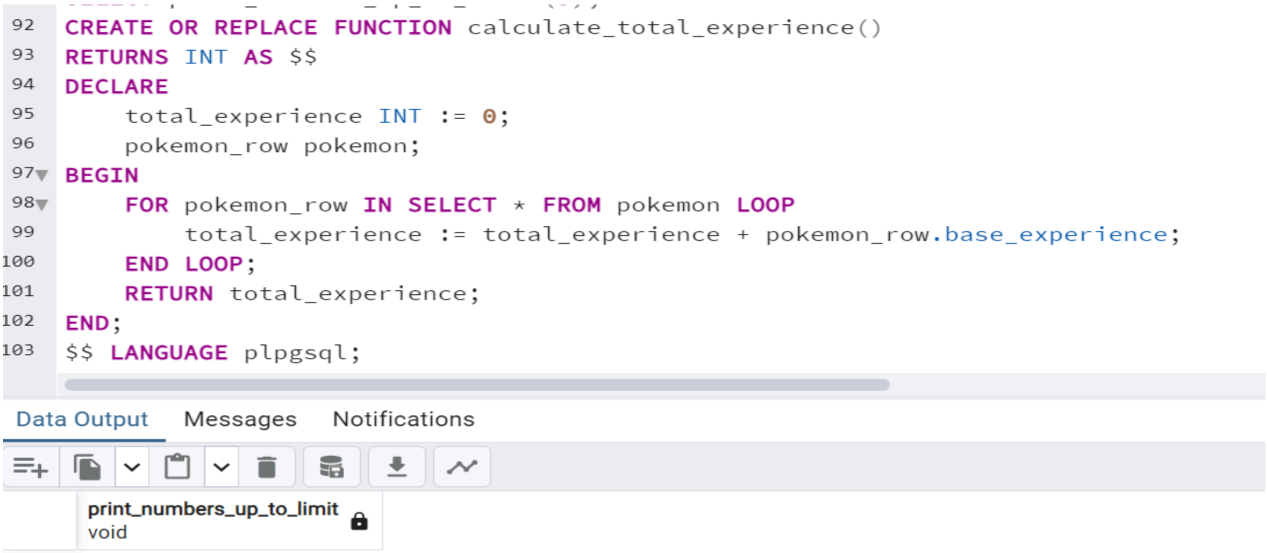
The function prints the number of pokemon up to the specific limit for each pokemon in the database.



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

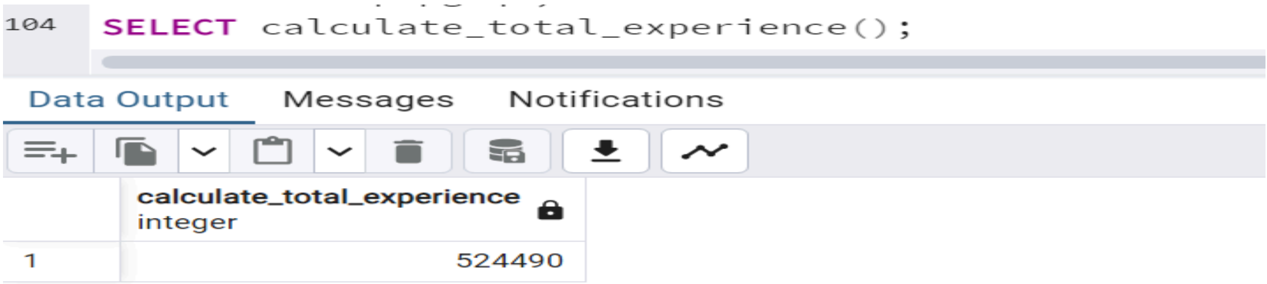
	print_numbers_up_to_limit void
1	

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;
```

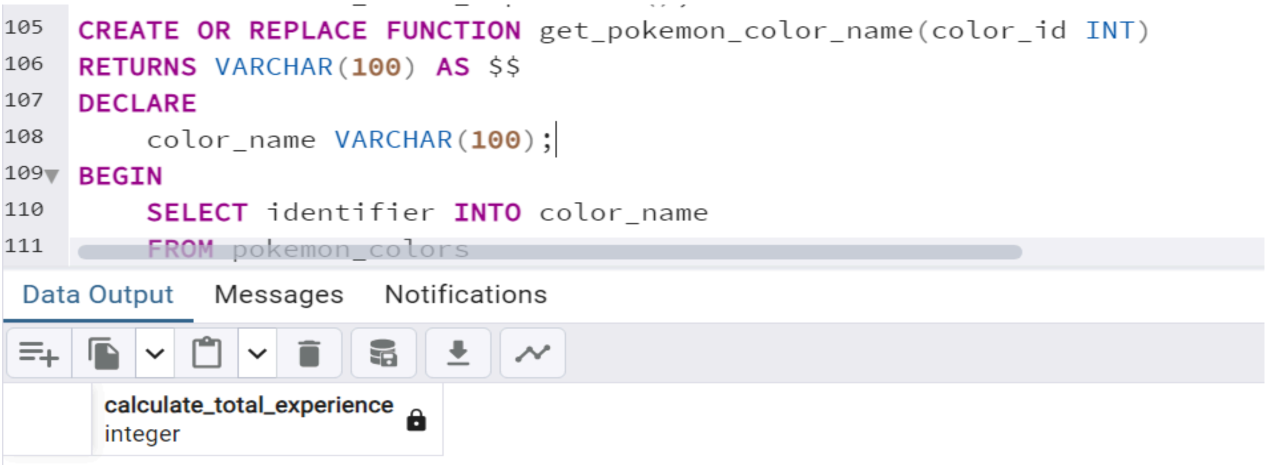
	print_numbers_up_to_limit void



```
SELECT calculate_total_experience();
```

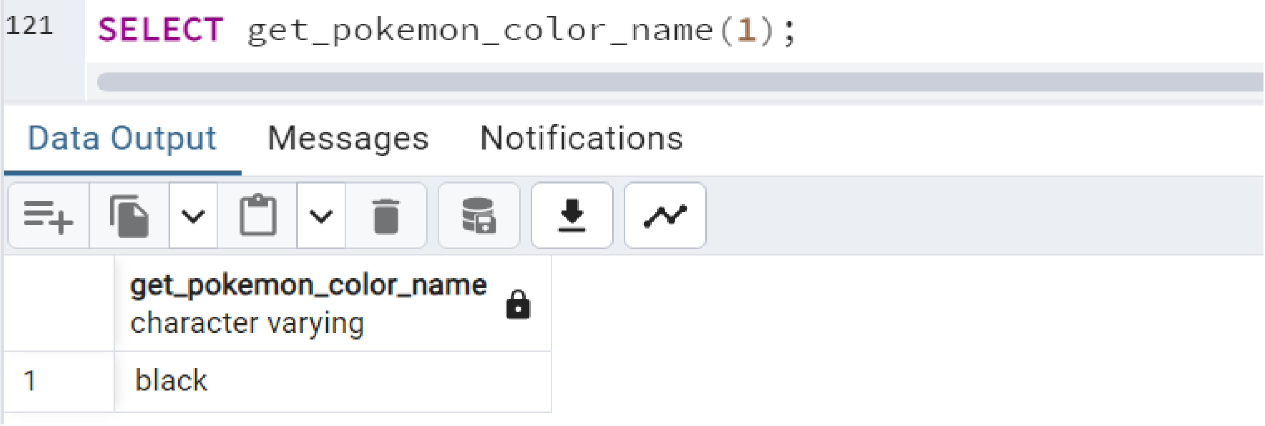
	calculate_total_experience integer
1	524490

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



```sql
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
```

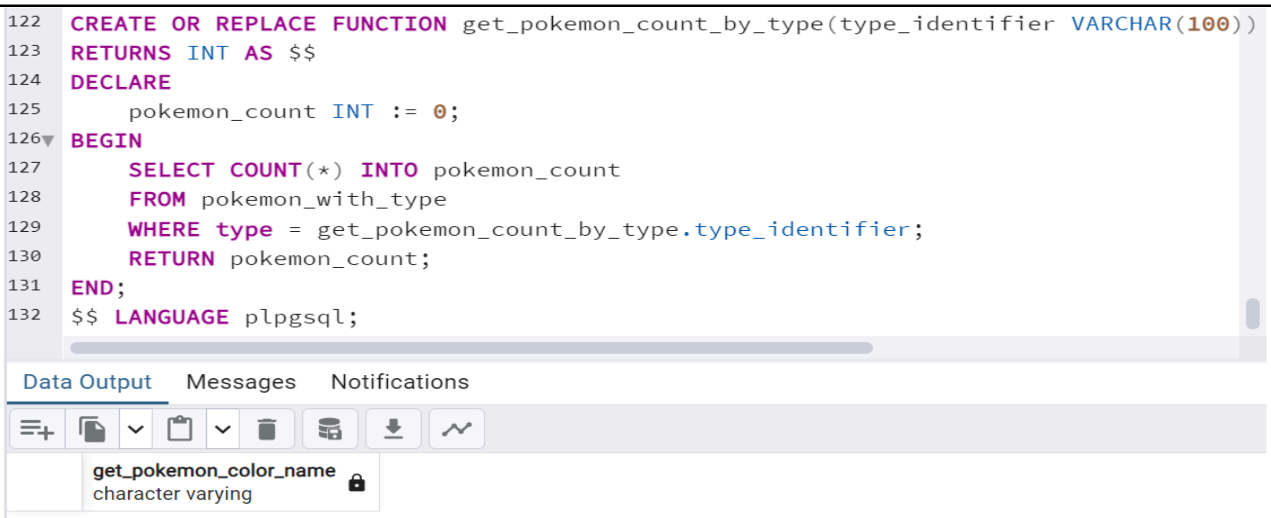
```
calculate_total_experience
integer
```



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

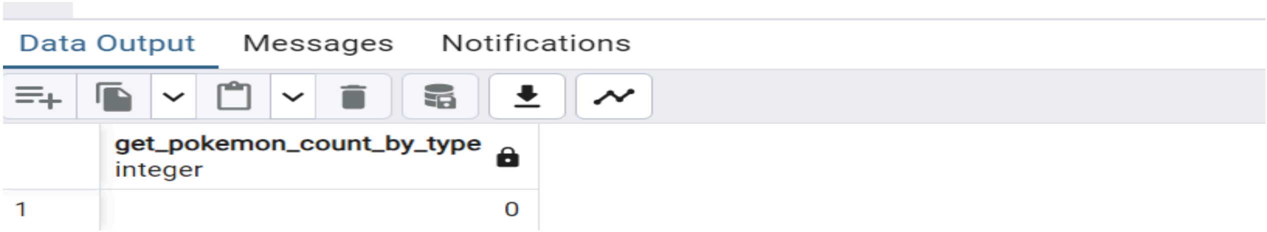
```
   get_pokemon_color_name
   character varying
1  black
```

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.



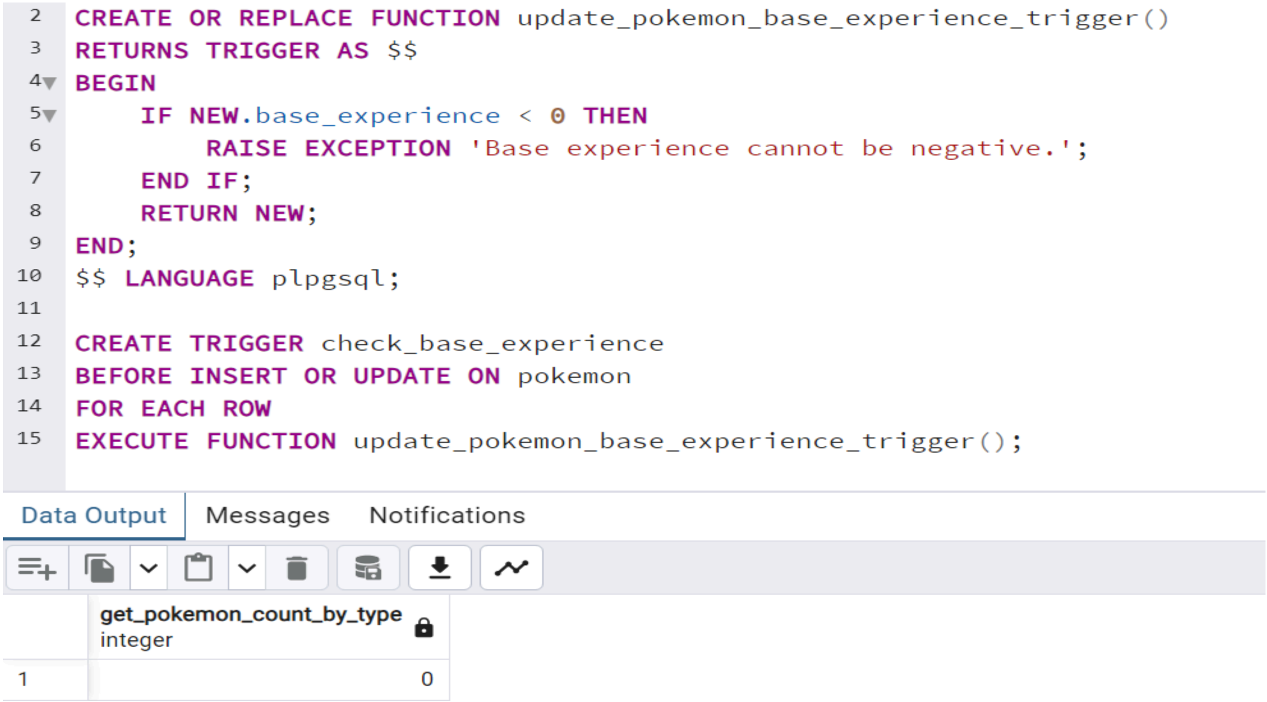
```
CREATE OR REPLACE FUNCTION get_pokemon_count_by_type(type_identifier VARCHAR(100))
RETURNS INT AS $$
DECLARE
    pokemon_count INT := 0;
BEGIN
    SELECT COUNT(*) INTO pokemon_count
    FROM pokemon_with_type
    WHERE type = get_pokemon_count_by_type.type_identifier;
    RETURN pokemon_count;
END;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

```
get_pokemon_color_name
character varying
```



```
   get_pokemon_count_by_type
   integer
1                          0
```

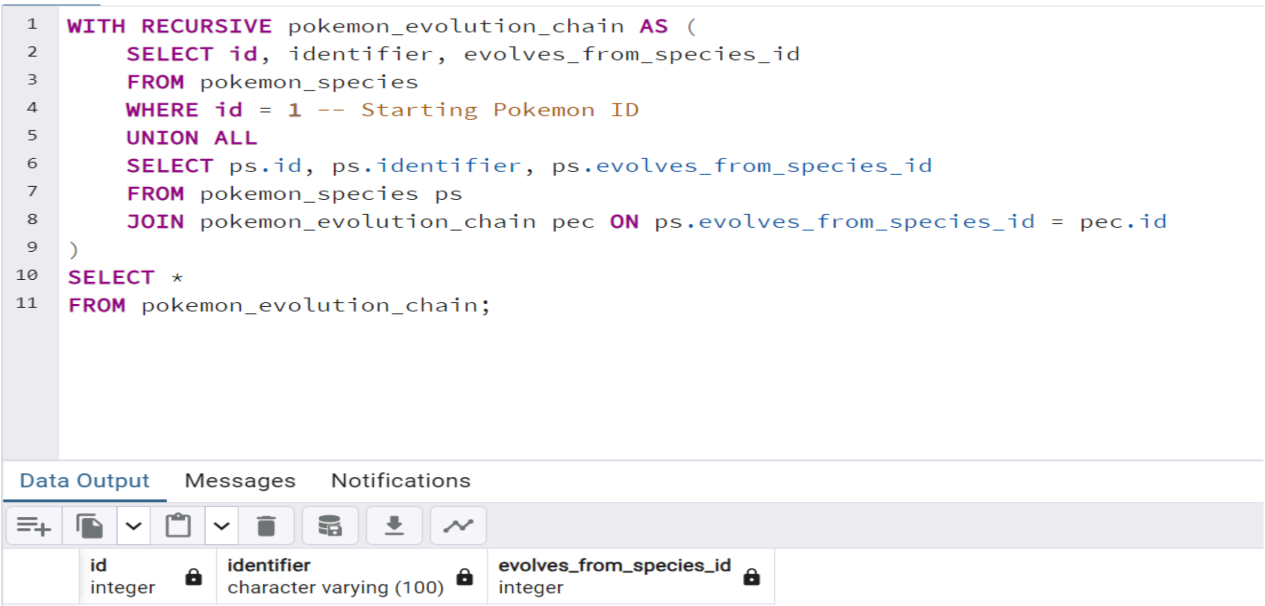
Task 6 : The function checks for the base experience of the pokemon, whether their base experience is negative or positive before updating or inserting into the database.



```
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();
```

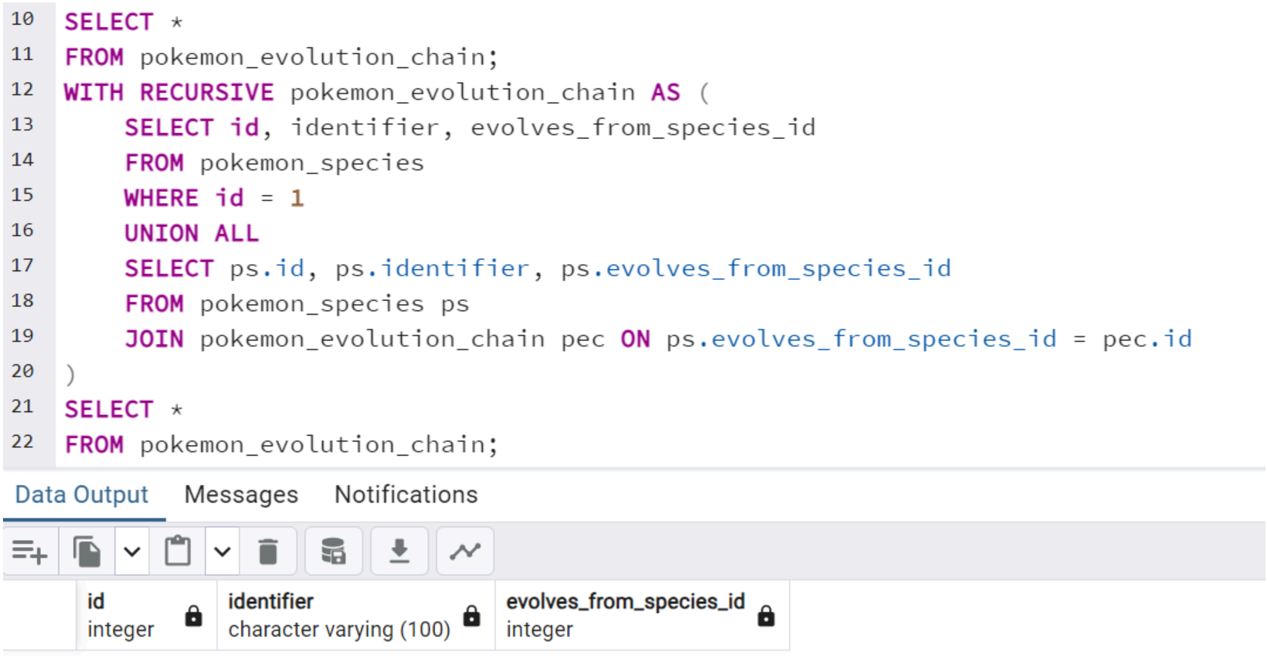
```
   get_pokemon_count_by_type
   integer
1                          0
```

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.



```
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;
```

```
id        identifier               evolves_from_species_id
integer   character varying (100)  integer
```



```
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;
```

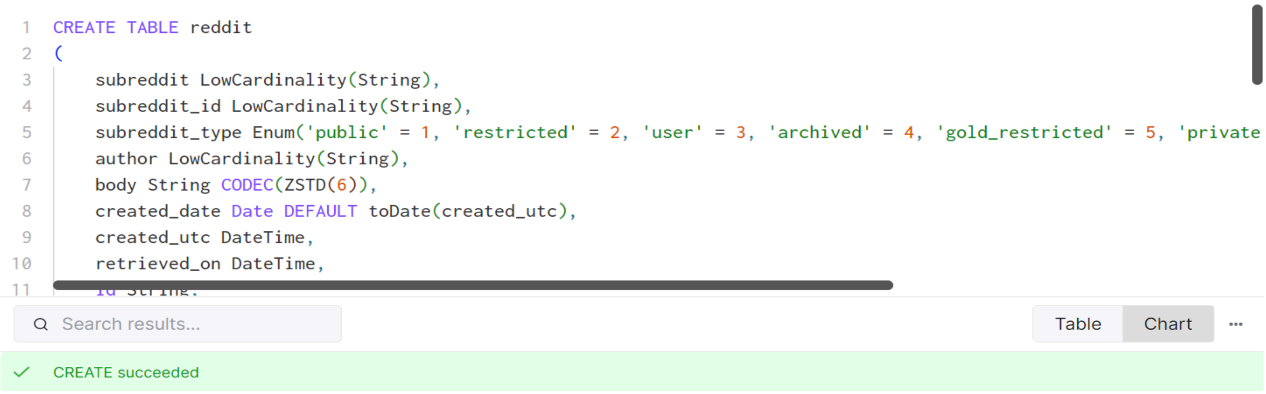
```
id        identifier               evolves_from_species_id
integer   character varying (100)  integer
```

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.

Topic : Practical 5 (reddit comment)

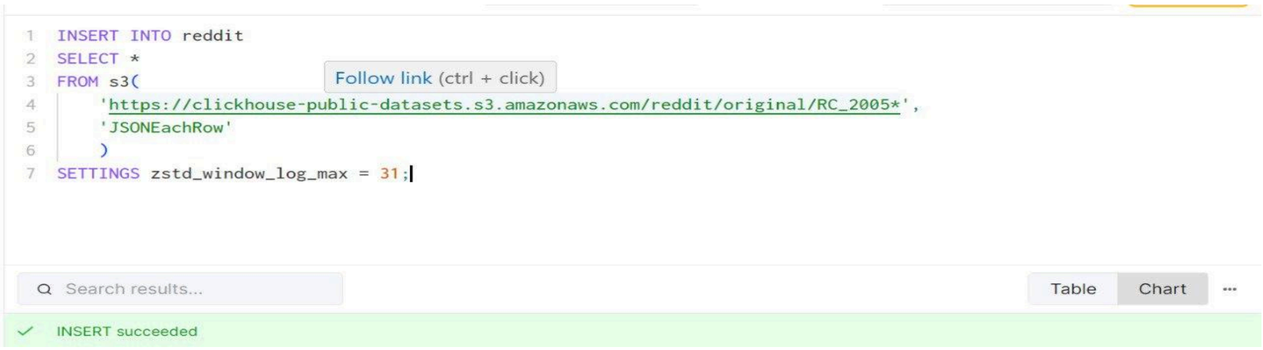
1 : Creating the table.



```
CREATE TABLE reddit
(
    subreddit LowCardinality(String),
    subreddit_id LowCardinality(String),
    subreddit_type Enum('public' = 1, 'restricted' = 2, 'user' = 3, 'archived' = 4, 'gold_restricted' = 5, 'private
    author LowCardinality(String),
    body String CODEC(ZSTD(6)),
    created_date Date DEFAULT toDate(created_utc),
    created_utc DateTime,
    retrieved_on DateTime,
    id String,
```

CREATE succeeded

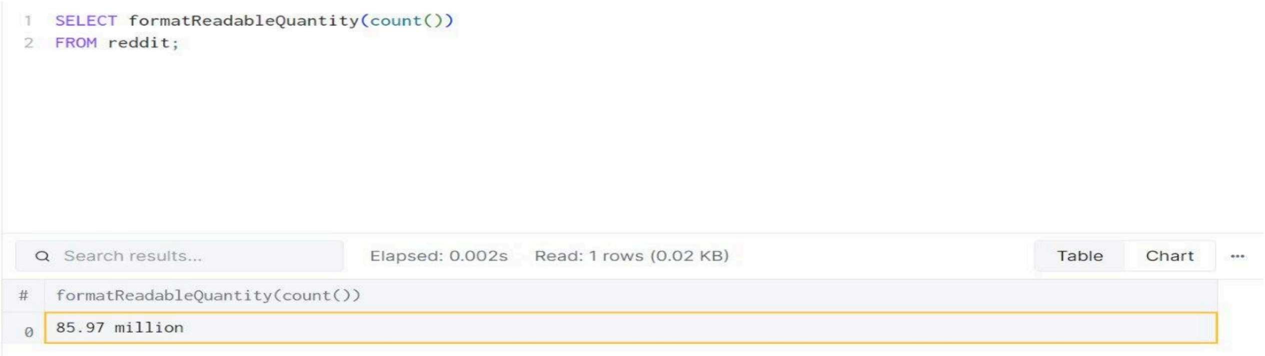
2 : In this task we are importing data directly from the S3.



```
INSERT INTO reddit
SELECT *
FROM s3(
    'https://clickhouse-public-datasets.s3.amazonaws.com/reddit/original/RC_2005*',
    'JSONEachRow'
    )
SETTINGS zstd_window_log_max = 31;
```

INSERT succeeded

3 : In this task the function is to format the numbers in a human readable format.

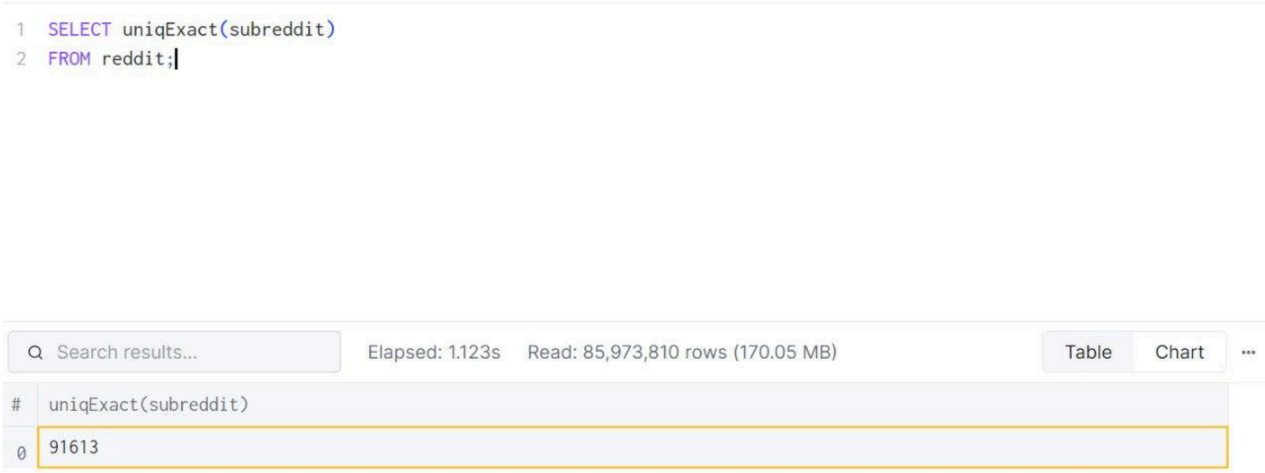


```
SELECT formatReadableQuantity(count())
FROM reddit;
```

Elapsed: 0.002s   Read: 1 rows (0.02 KB)

#	formatReadableQuantity(count())
0	85.97 million

4 : The function in this task is to count the exact number of distinct values in a column.

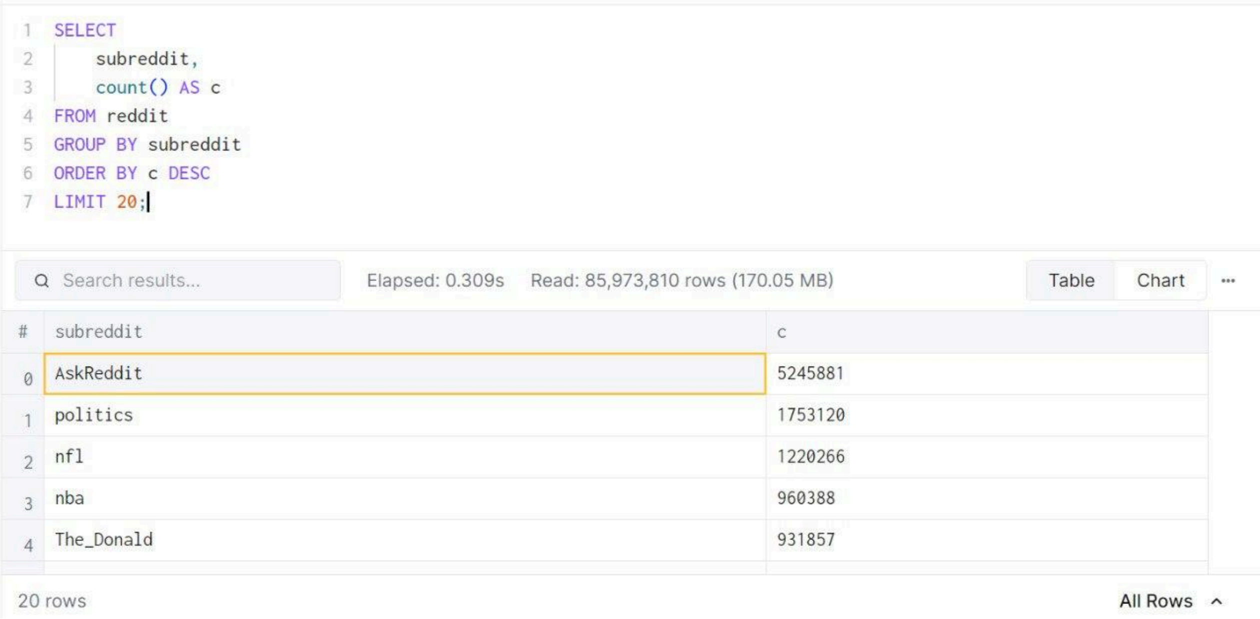


```
SELECT uniqExact(subreddit)
FROM reddit;
```

Search results... Elapsed: 1.123s Read: 85,973,810 rows (170.05 MB) Table Chart

#	uniqExact(subreddit)
0	91613

5 : In this task the function is to count the number of rows in each group.



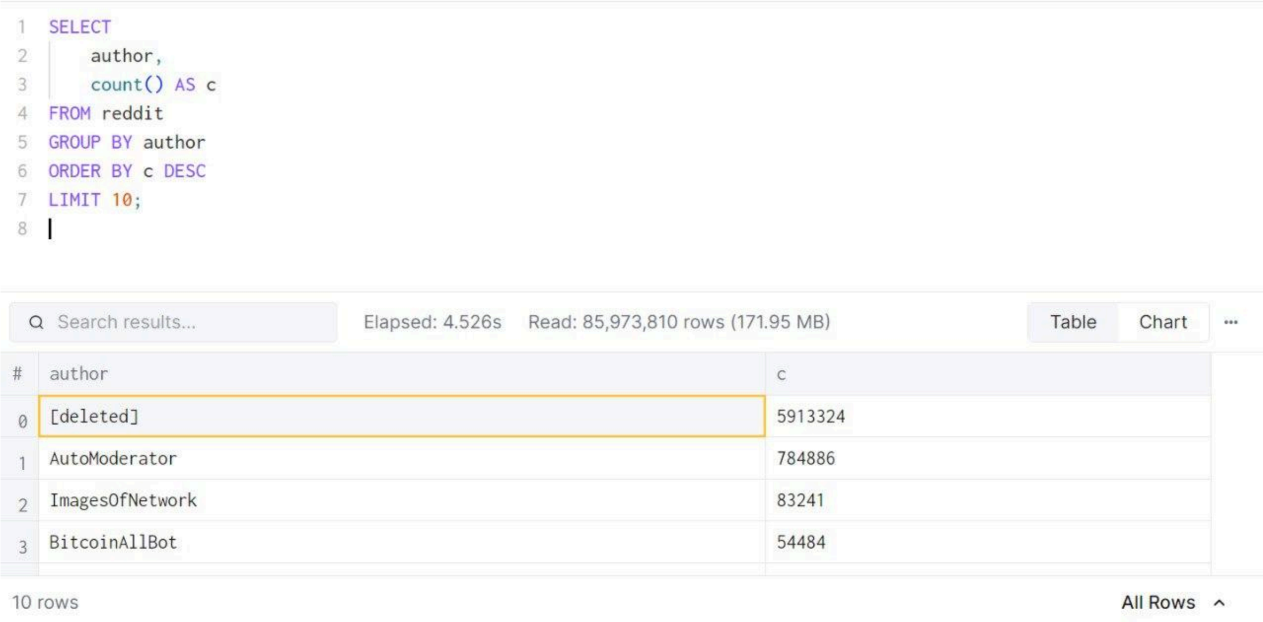
```
SELECT
    subreddit,
    count() AS c
FROM reddit
GROUP BY subreddit
ORDER BY c DESC
LIMIT 20;
```

Search results... Elapsed: 0.309s Read: 85,973,810 rows (170.05 MB) Table Chart

#	subreddit	c
0	AskReddit	5245881
1	politics	1753120
2	nfl	1220266
3	nba	960388
4	The_Donald	931857

20 rows All Rows

6 : The function in this is to count the number of rows in each group.



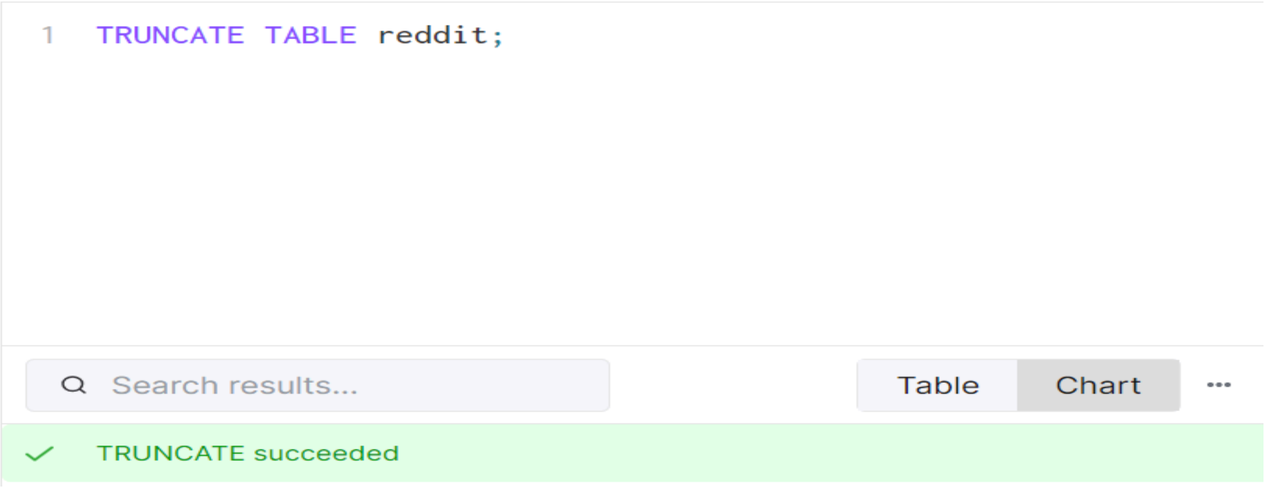
```sql
SELECT
    author,
    count() AS c
FROM reddit
GROUP BY author
ORDER BY c DESC
LIMIT 10;
```

Elapsed: 4.526s Read: 85,973,810 rows (171.95 MB)

#	author	c
0	[deleted]	5913324
1	AutoModerator	784886
2	ImagesOfNetwork	83241
3	BitcoinAllBot	54484

10 rows

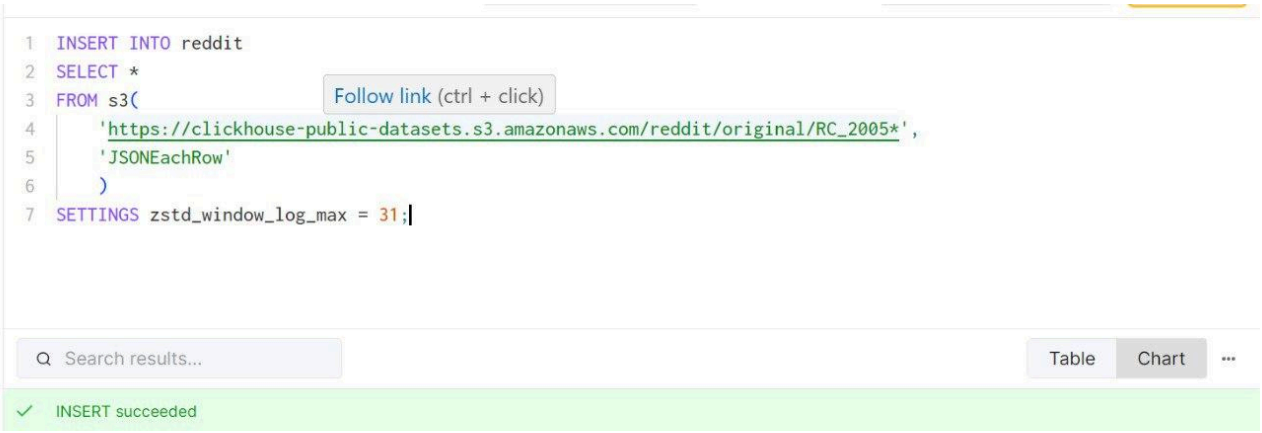
7 : The function is used to delete all the rows from the reddit table in the database.



```sql
TRUNCATE TABLE reddit;
```

✓ TRUNCATE succeeded

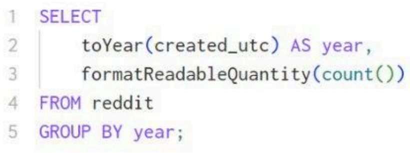
8 : The function is used to import the data directly from S3. It imports the data from 2005.



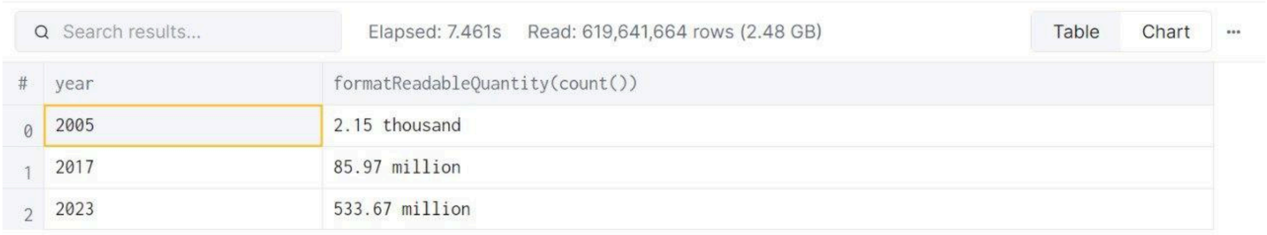
```
INSERT INTO reddit
SELECT *
FROM s3(
    'https://clickhouse-public-datasets.s3.amazonaws.com/reddit/original/RC_2005*',
    'JSONEachRow'
    )
SETTINGS zstd_window_log_max = 31;
```

INSERT succeeded

9 : The function is used to extract the year from a date and format the number in a human readable form.



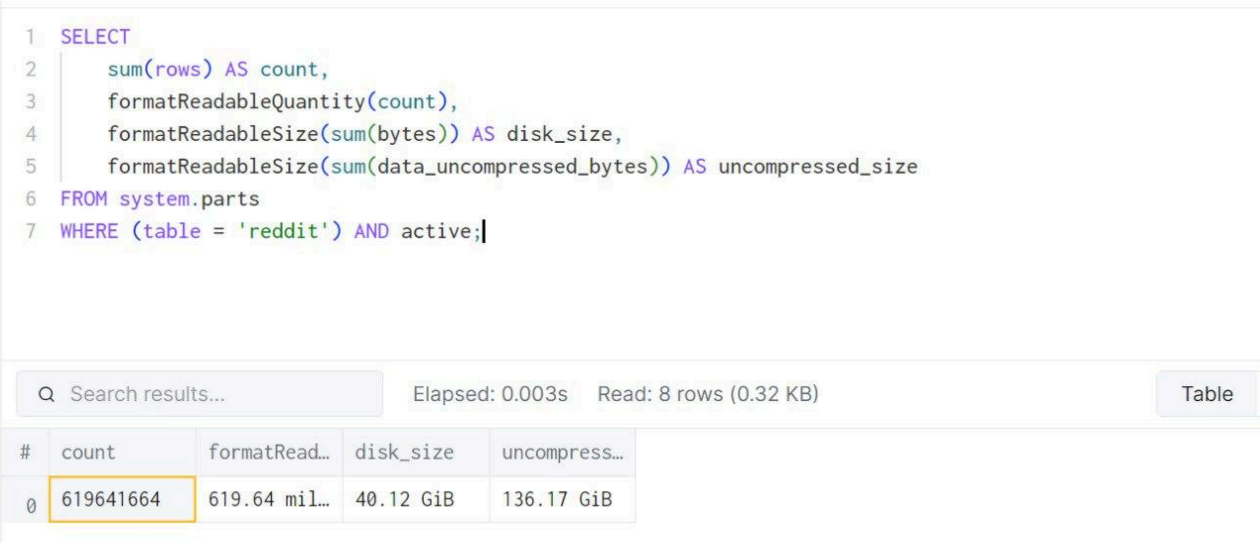
```
SELECT
    toYear(created_utc) AS year,
    formatReadableQuantity(count())
FROM reddit
GROUP BY year;
```



Elapsed: 7.461s   Read: 619,641,664 rows (2.48 GB)

#	year	formatReadableQuantity(count())
0	2005	2.15 thousand
1	2017	85.97 million
2	2023	533.67 million

10 : Let's see how many rows were inserted and how much disk space the table is using:

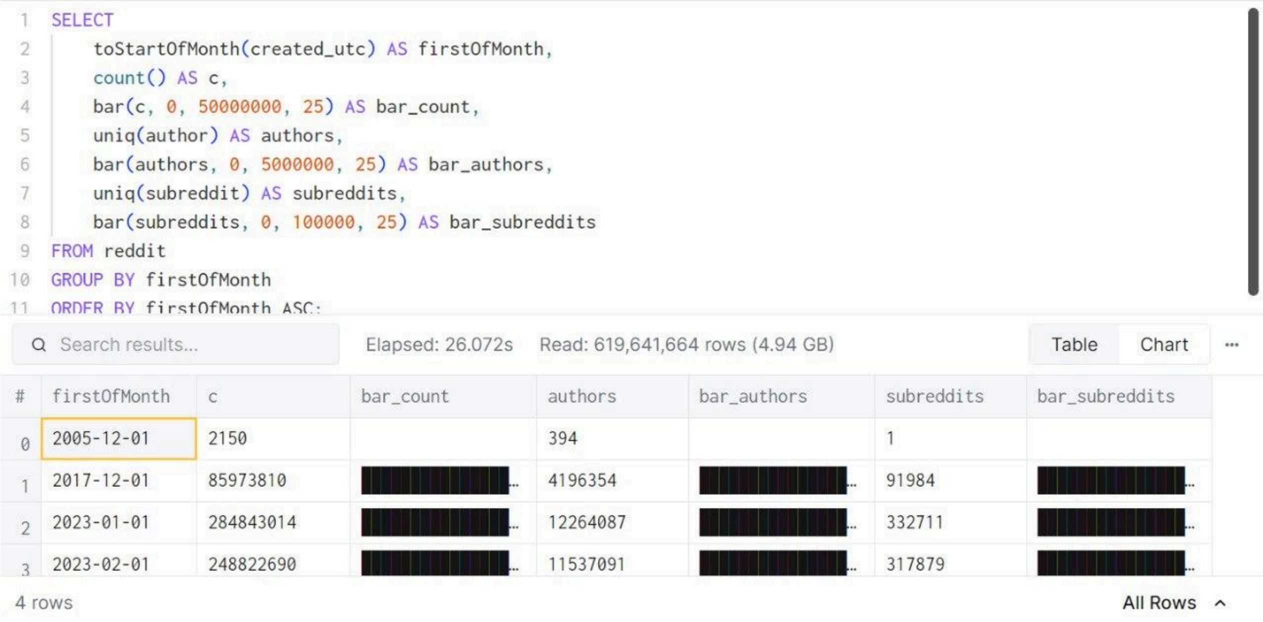


```sql
SELECT
    sum(rows) AS count,
    formatReadableQuantity(count),
    formatReadableSize(sum(bytes)) AS disk_size,
    formatReadableSize(sum(data_uncompressed_bytes)) AS uncompressed_size
FROM system.parts
WHERE (table = 'reddit') AND active;
```

Elapsed: 0.003s Read: 8 rows (0.32 KB)

#	count	formatRead...	disk_size	uncompress...
0	619641664	619.64 mil...	40.12 GiB	136.17 GiB

11 : The function is used to group the data by the starting of the month and count the total number of rows, creates a bar chart of the count, finds the unique number of the authors, create a bar chart of the unique authors, finds the unique number of subreddits and creates a bar chart of the unique subreddits.



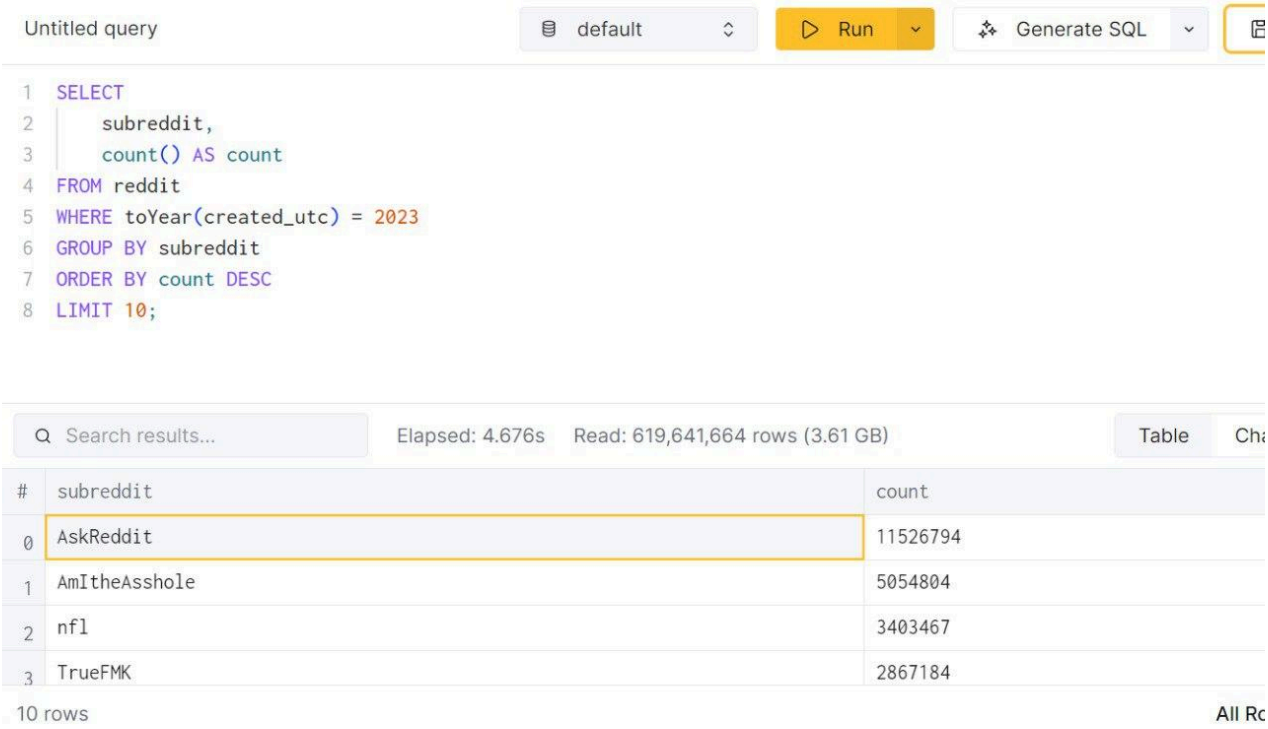
```sql
SELECT
    toStartOfMonth(created_utc) AS firstOfMonth,
    count() AS c,
    bar(c, 0, 50000000, 25) AS bar_count,
    uniq(author) AS authors,
    bar(authors, 0, 5000000, 25) AS bar_authors,
    uniq(subreddit) AS subreddits,
    bar(subreddits, 0, 100000, 25) AS bar_subreddits
FROM reddit
GROUP BY firstOfMonth
ORDER BY firstOfMonth ASC;
```

Elapsed: 26.072s Read: 619,641,664 rows (4.94 GB)

#	firstOfMonth	c	bar_count	authors	bar_authors	subreddits	bar_subreddits
0	2005-12-01	2150		394		1	
1	2017-12-01	85973810	████████████████...	4196354	████████████████...	91984	████████████████...
2	2023-01-01	284843014	████████████████...	12264087	████████████████...	332711	████████████████...
3	2023-02-01	248822690	████████████████...	11537091	████████████████...	317879	████████████████...

4 rows

12 : The function is used to extract the year and function to count the number of rows in each group.



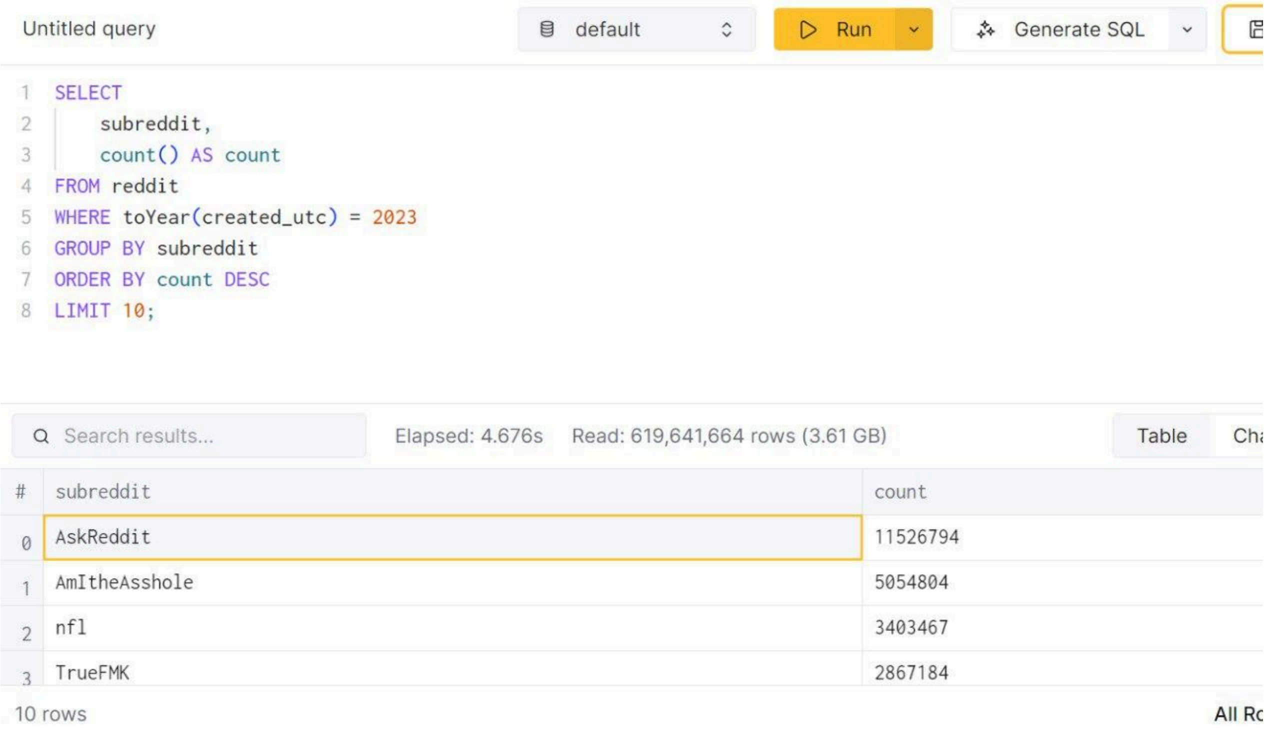
```sql
SELECT
    subreddit,
    count() AS count
FROM reddit
WHERE toYear(created_utc) = 2023
GROUP BY subreddit
ORDER BY count DESC
LIMIT 10;
```

Elapsed: 4.676s Read: 619,641,664 rows (3.61 GB)

#	subreddit	count
0	AskReddit	11526794
1	AmItheAsshole	5054804
2	nfl	3403467
3	TrueFMK	2867184

10 rows

13 : The function is used to extract the year from a date and a function to count the number of rows in each group.



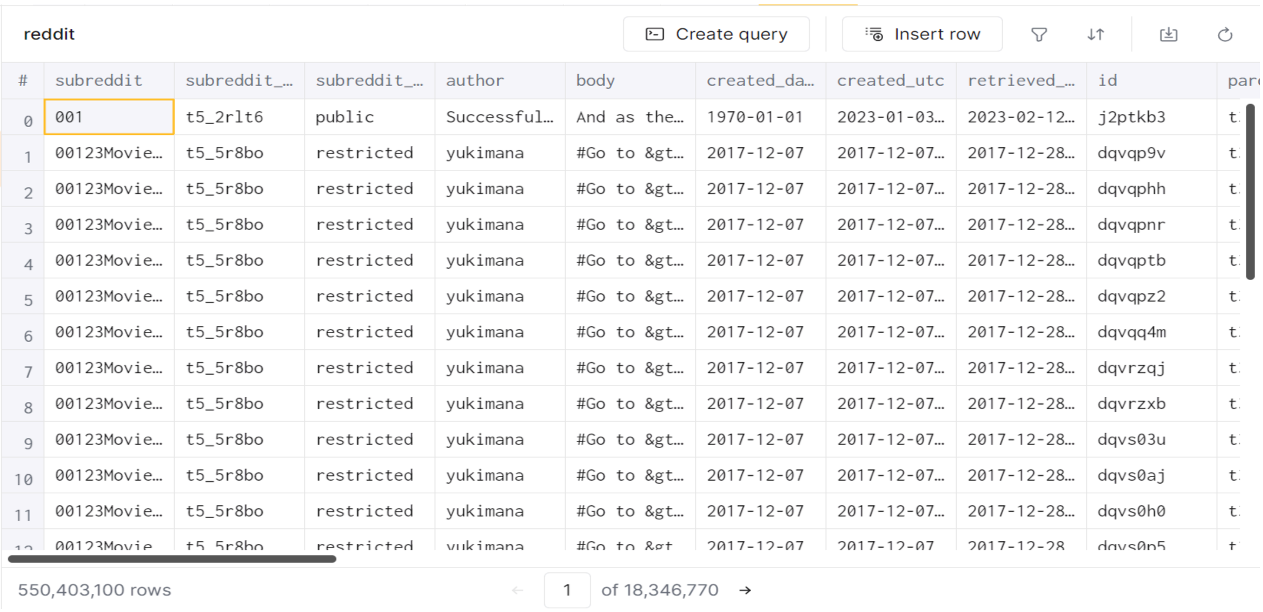
```sql
SELECT
    subreddit,
    count() AS count
FROM reddit
WHERE toYear(created_utc) = 2023
GROUP BY subreddit
ORDER BY count DESC
LIMIT 10;
```

Elapsed: 4.676s   Read: 619,641,664 rows (3.61 GB)

#	subreddit	count
0	AskReddit	11526794
1	AmItheAsshole	5054804
2	nfl	3403467
3	TrueFMK	2867184

10 rows

14 : The function is used to extract the data start of the quarter from a date and a function to find the position of a substring in a string.



#	subreddit	subreddit_...	subreddit_...	author	body	created_da...	created_utc	retrieved_...	id	par
0	001	t5_2rlt6	public	Successful...	And as the...	1970-01-01	2023-01-03...	2023-02-12...	j2ptkb3	t
1	00123Movie...	t5_5r8bo	restricted	yukimana	#Go to &gt...	2017-12-07	2017-12-07...	2017-12-28...	dqvqp9v	t
2	00123Movie...	t5_5r8bo	restricted	yukimana	#Go to &gt...	2017-12-07	2017-12-07...	2017-12-28...	dqvqphh	t
3	00123Movie...	t5_5r8bo	restricted	yukimana	#Go to &gt...	2017-12-07	2017-12-07...	2017-12-28...	dqvqpnr	t
4	00123Movie...	t5_5r8bo	restricted	yukimana	#Go to &gt...	2017-12-07	2017-12-07...	2017-12-28...	dqvqptb	t
5	00123Movie...	t5_5r8bo	restricted	yukimana	#Go to &gt...	2017-12-07	2017-12-07...	2017-12-28...	dqvqpz2	t
6	00123Movie...	t5_5r8bo	restricted	yukimana	#Go to &gt...	2017-12-07	2017-12-07...	2017-12-28...	dqvqq4m	t
7	00123Movie...	t5_5r8bo	restricted	yukimana	#Go to &gt...	2017-12-07	2017-12-07...	2017-12-28...	dqvrzqj	t
8	00123Movie...	t5_5r8bo	restricted	yukimana	#Go to &gt...	2017-12-07	2017-12-07...	2017-12-28...	dqvrzxb	t
9	00123Movie...	t5_5r8bo	restricted	yukimana	#Go to &gt...	2017-12-07	2017-12-07...	2017-12-28...	dqvs03u	t
10	00123Movie...	t5_5r8bo	restricted	yukimana	#Go to &gt...	2017-12-07	2017-12-07...	2017-12-28...	dqvs0aj	t
11	00123Movie...	t5_5r8bo	restricted	yukimana	#Go to &gt...	2017-12-07	2017-12-07...	2017-12-28...	dqvs0h0	t
12	00123Movie	t5_5r8bo	restricted	yukimana	#Go to &gt	2017-12-07	2017-12-07	2017-12-28	dqvs0n5	t

550,403,100 rows

Conclusion :

The task helps us to understand the techniques and application of data manipulation and analysis of dataset and it also includes creating tables, importing data, formatting numbers for human readability, counting distinct values in columns and grouping the data count within each group. So the powerful tool for the analysis includes Count(distinct column name) and Group by clause that allows functions to count the distinct values within a categorical column to understand the uniqueness and diversity of the database and it also has a function to handle the null values in the sql queries.