Managing Duplicate Records in Normalized Database Using

PostgreSQL

Task Summary

Task Objectives:

- Identify duplicate records within the PostgreSQL database.
- Develop SQL queries or scripts to consolidate duplicate records based on predefined criteria.
- Utilize PostgreSQL features such as DISTINCT, GROUP BY, and window functions to detect and manage duplicates efficiently.
- Document the duplicate management process for future reference and training purposes.

Key Takeaways:

- Duplicate records can lead to inconsistencies and inaccuracies in data analysis and reporting.
- Duplicate records consume additional storage space, impacting database performance and scalability.
- Ensure you have a backup before updating the table particularly when deleting multiple records.
- Documentation is invaluable, particularly when facing errors, syntax lapses, or encountering new concepts beyond the course material.
- In PostgreSQL, there are many solutions available for various problems. Your responsibility is to explore and document them accordingly for future reference.

Cris Bailon M. Camacho
March 21, 2024

categories TABLE

1

```sql
--BASIC INFORMATION OF categories TABLE
SELECT column_name, data_type
FROM INFORMATION_SCHEMA.COLUMNS
WHERE table_name = 'categories';
```

column_name name	data_type character varying
category_id	integer
name	text

2

```sql
--CHECKING FOR DUPLICATE RECORDS IN categories TABLE
SELECT name, COUNT(*)
FROM categories
GROUP BY name
HAVING COUNT(*) >1;
```

name text	count bigint
Electronics	2

3

```sql
/*DUPLICATE RECORDS WERE FOUND
IN categories TABLE*/
SELECT *
FROM categories
WHERE name = 'Electronics';
```

category_id integer	name text
37	Electronics
13	Electronics

4

```sql
/*DELETE ONE OF THE DUPLICATED RECORDS, THEN
UPDATE THE products TABLE AS IT IS RELATED TO IT*/
DELETE
FROM categories
WHERE category_id = 37;

--UPDATING THE products TABLE
UPDATE products
SET category_id = 13
WHERE category_id = 37;
```

products TABLE

1

```sql
--BASIC INFORMATION OF products TABLE
SELECT column_name, data_type
FROM INFORMATION_SCHEMA.COLUMNS
WHERE table_name = 'products'
ORDER BY ordinal_position;
```

column_name name	data_type character varying
product_id	integer
name	text
price	numeric
image_url	text
category_id	integer

2

```sql
--CHECKING FOR DUPLICATE RECORDS IN products TABLE
SELECT
    name,
    price,
    COUNT(*)
FROM products
GROUP BY name,price
HAVING COUNT(*) >1
```

NO DUPLICATE RECORDS

customers TABLE

1

```sql
--CHECKING FOR DUPLICATE RECORDS IN customers TABLE
SELECT
    first_name, last_name, email, password, city,
    country, segment, state, street, zip_code
FROM customers
GROUP BY
    first_name, last_name, email, password, city,
    country, segment, state, street, zip_code
HAVING COUNT(*) >1
```

first_name text	last_name text	email text	password character varying (10)	city text	country text	segment text	state text	street text	zip_code text
Mary	Smith	XXXXXXXXX	XXXXXXXXX	Caguas	Puerto Rico	Home Office	PR	4896 Honey Wood	725
Mary	Smith	XXXXXXXXX	XXXXXXXXX	Caguas	Puerto Rico	Consumer	PR	3544 Gentle Concession	725
Mary	Smith	XXXXXXXXX	XXXXXXXXX	Caguas	Puerto Rico	Consumer	PR	7446 Silent Vista	725
Mary	Smith	XXXXXXXXX	XXXXXXXXX	Caguas	Puerto Rico	Consumer	PR	1736 Clear Gate	725
Mary	Smith	XXXXXXXXX	XXXXXXXXX	Caguas	Puerto Rico	Consumer	PR	4839 Lazy View	725
Mary	Smith	XXXXXXXXX	XXXXXXXXX	Caguas	Puerto Rico	Consumer	PR	5406 Misty Timber End	725
Mary	Smith	XXXXXXXXX	XXXXXXXXX	Caguas	Puerto Rico	Corporate	PR	654 Gentle Acres	725
Mary	Smith	XXXXXXXXX	XXXXXXXXX	Caguas	Puerto Rico	Consumer	PR	190 Heather View	725
Mary	Smith	XXXXXXXXX	XXXXXXXXX	Caguas	Puerto Rico	Corporate	PR	7368 Colonial Range	725
Mary	Smith	XXXXXXXXX	XXXXXXXXX	Caguas	Puerto Rico	Consumer	PR	7274 Middle Wynd	725
Mary	Smith	XXXXXXXXX	XXXXXXXXX	Caguas	Puerto Rico	Corporate	PR	3867 Bright Zephyr Ledge	725

2

```sql
SELECT * --VIEWING FOR DUPLICATE RECORDS IN customers TABLE--STEP 1
FROM customers
WHERE (first_name, last_name, email, password, city,
    country, segment, state, street, zip_code) IN
        (
        SELECT
            first_name, last_name, email, password, city,
            country, segment, state, street, zip_code
        FROM customers
        GROUP BY
            first_name, last_name, email, password, city,
            country, segment, state, street, zip_code
        HAVING COUNT(*) >1
        )
ORDER BY street, segment DESC, customer_id
        )
```

customer_id integer	first_name text	last_name text	email text	password character varying (10)	city text	country text	segment text	state text	street text	zip_code text
5286	Mary	Smith	XXXXXXXXX	XXXXXXXXX	Caguas	Puerto Rico	Consumer	PR	1736 Clear Gate	725
12024	Mary	Smith	XXXXXXXXX	XXXXXXXXX	Caguas	Puerto Rico	Consumer	PR	1736 Clear Gate	725
1495	Mary	Smith	XXXXXXXXX	XXXXXXXXX	Caguas	Puerto Rico	Consumer	PR	190 Heather View	725
9619	Mary	Smith	XXXXXXXXX	XXXXXXXXX	Caguas	Puerto Rico	Consumer	PR	190 Heather View	725
3765	Mary	Smith	XXXXXXXXX	XXXXXXXXX	Caguas	Puerto Rico	Consumer	PR	3544 Gentle Concession	725
4243	Mary	Smith	XXXXXXXXX	XXXXXXXXX	Caguas	Puerto Rico	Consumer	PR	3544 Gentle Concession	725
2083	Mary	Smith	XXXXXXXXX	XXXXXXXXX	Caguas	Puerto Rico	Corporate	PR	3867 Bright Zephyr Ledge	725
9358	Mary	Smith	XXXXXXXXX	XXXXXXXXX	Caguas	Puerto Rico	Corporate	PR	3867 Bright Zephyr Ledge	725

3

```sql
WITH duplicate AS( --USE CTE TO DISPLAY DUPLICATE RECORDS --STEP 2
    SELECT * --VIEWING FOR DUPLICATE RECORDS IN customers TABLE--STEP 1
    FROM customers
    WHERE (first_name, last_name, email, password, city,
            country, segment, state, street, zip_code) IN
            (
                SELECT
                first_name, last_name, email, password, city,
                country, segment, state, street, zip_code
                FROM customers
                GROUP BY
                first_name, last_name, email, password, city,
                country, segment, state, street, zip_code
                HAVING COUNT(*) >1
            )
    ORDER BY street, segment DESC, customer_id
        )
SELECT --SEPARATE ORIGINAL IDs TO DUPLICATE IDs --STEP 3
    MAX(CASE WHEN row_num % 2 <> 0 THEN customer_id END) AS duplicated_id,
    MAX(CASE WHEN row_num % 2 = 0 THEN customer_id END) AS original_id
FROM (
    SELECT customer_id,
    ROW_NUMBER() OVER() AS row_num
    FROM duplicate
    ) AS numbered_data
GROUP BY CEIL(row_num / 2.0)
ORDER BY MIN(row_num);
```

duplicated_id integer	original_id integer
5286	12024
1495	9619
3765	4243
2083	9358
2005	7674
644	5486
1493	1722
3861	6840
344	1851
1954	5498
11557	11866

customer_id integer	first_name text	last_name text	email text	password character varying (10)	city text	country text	segment text	state text	street text	zip_code text
5286	Mary	Smith	XXXXXXXXX	XXXXXXXXX	Caguas	Puerto Rico	Consumer	PR	1736 Clear Gate	725
12024	Mary	Smith	XXXXXXXXX	XXXXXXXXX	Caguas	Puerto Rico	Consumer	PR	1736 Clear Gate	725
1495	Mary	Smith	XXXXXXXXX	XXXXXXXXX	Caguas	Puerto Rico	Consumer	PR	190 Heather View	725
9619	Mary	Smith	XXXXXXXXX	XXXXXXXXX	Caguas	Puerto Rico	Consumer	PR	190 Heather View	725
3765	Mary	Smith	XXXXXXXXX	XXXXXXXXX	Caguas	Puerto Rico	Consumer	PR	3544 Gentle Concession	725
4243	Mary	Smith	XXXXXXXXX	XXXXXXXXX	Caguas	Puerto Rico	Consumer	PR	3544 Gentle Concession	725
2083	Mary	Smith	XXXXXXXXX	XXXXXXXXX	Caguas	Puerto Rico	Corporate	PR	3867 Bright Zephyr Ledge	725
9358	Mary	Smith	XXXXXXXXX	XXXXXXXXX	Caguas	Puerto Rico	Corporate	PR	3867 Bright Zephyr Ledge	725

duplicated_id integer	original_id integer
5286	12024
1495	9619
3765	4243
2083	9358
2005	7674
644	5486
1493	1722
3861	6840
344	1851
1954	5498
11557	11866

Make sure the results are accurate.

Note: When separating original IDs to duplicate IDs, ensure that the count of duplicate records is exactly 2.

4

```sql
--DELETE DUPLICATE RECORDS IN customers TABLE
DELETE
FROM customers AS dup_id
USING customers AS dist_id
WHERE dup_id < dist_id
    AND dup_id.first_name = dist_id.first_name
    AND dup_id.last_name = dist_id.last_name
    AND dup_id.email = dist_id.email
    AND dup_id.password = dist_id.password
    AND dup_id.city = dist_id.city
    AND dup_id.country = dist_id.country
    AND dup_id.segment = dist_id.segment
    AND dup_id.state = dist_id.state
    AND dup_id.street = dist_id.street
    AND dup_id.zip_code = dist_id.zip_code
RETURNING dup_id.customer_id AS deleted_customer_id
```

deleted_customer_id integer
344
644
1493
1495
1954
2005
2083
3765
3861
5286
11557

5

```sql
--UPDATE THE fact_table TABLE AS IT IS RELATED TO customers_table
UPDATE  fact_table
SET customer_id =
    CASE
        WHEN customer_id = 5286 THEN 12024
        WHEN customer_id = 1495 THEN 9619
        WHEN customer_id = 3765 THEN 4243
        WHEN customer_id = 2083 THEN 9358
        WHEN customer_id = 2005 THEN 7674
        WHEN customer_id = 644 THEN 5486
        WHEN customer_id = 1493 THEN 1722
        WHEN customer_id = 3861 THEN 6840
        WHEN customer_id = 344 THEN 1851
        WHEN customer_id = 1954 THEN 5498
        WHEN customer_id = 11557 THEN 11866
        ELSE customer_id
    END;
```

6

```sql
--UPDATE THE order_locations TABLE AS IT IS RELATED TO customers_table
UPDATE order_locations
SET order_customer_id =
    CASE
        WHEN order_customer_id = 5286 THEN 12024
        WHEN order_customer_id = 1495 THEN 9619
        WHEN order_customer_id = 3765 THEN 4243
        WHEN order_customer_id = 2083 THEN 9358
        WHEN order_customer_id = 2005 THEN 7674
        WHEN order_customer_id = 644 THEN 5486
        WHEN order_customer_id = 1493 THEN 1722
        WHEN order_customer_id = 3861 THEN 6840
        WHEN order_customer_id = 344 THEN 1851
        WHEN order_customer_id = 1954 THEN 5498
        WHEN order_customer_id = 11557 THEN 11866
        ELSE order_customer_id
    END;
```

order_locations TABLE

1

```
--CHECKING DUPLICATE RECORDS IN order_locations TABLE
SELECT city,country,region, state, zip_code, order_customer_id, COUNT(*)
FROM order_locations
GROUP BY city,country,region, state, zip_code,order_customer_id
HAVING COUNT(*) > 1;
```

	city text	country text	region text	state text	zip_code integer	order_customer_id integer	count bigint
1	Dallas	Estados Unidos	US Center	Texas	75220	4574	2
2	Mexico City	Mexico	Central America	Distrito Federal	[null]	11037	2
3	Managua	Nicaragua	Central America	Managua	[null]	2995	2
4	Madrid	Espana	Southern Europe	Madrid	[null]	5914	2
5	Mexico City	Mexico	Central America	Distrito Federal	[null]	6305	2
6	Yakarta	Indonesia	Southeast Asia	Yakarta	[null]	9490	2

Total rows: 195 of 195 Query complete 00:00:01.361

2

```
--VALIDATING THE COUNT OF DUPLICATE RECORDS
SELECT city,country,region, state, zip_code, order_customer_id, COUNT(*)
FROM order_locations
GROUP BY city,country,region, state, zip_code,order_customer_id
HAVING COUNT(*) > 2;
```

city text	country text	region text	state text	zip_code integer	order_customer_id integer	count bigint
San Salvador	El Salvador	Central America	San Salvador	0	1524	3

3

```
SELECT * --VIEWING FOR DUPLICATE RECORDS IN order_locations TABLE--STEP 1
FROM order_locations
WHERE (city, country, region, state, zip_code, order_customer_id) IN (
    SELECT
        city, country, region, state, zip_code,order_customer_id
    FROM order_locations
    GROUP BY
        city, country, region, state, zip_code,order_customer_id
    HAVING COUNT(*) >1)
ORDER BY city, country, location_id DESC)
```

location_id integer	city text	country text	region text	state text	zip_code integer	order_customer_id integer
7955	San Salvador	El Salvador	Central America	San Salvador	0	1524
60350	San Salvador	El Salvador	Central America	San Salvador	0	1524
5189	San Salvador	El Salvador	Central America	San Salvador	0	1524

4

```
DELETE
FROM order_locations
WHERE location_id = 5189
```

```
--UPDATE fact_table AS IT IS RELATED TO IT
UPDATE fact_table
SET order_item_id = 603500
WHERE order_item_id = 5189;

--UPDATE order_items AS IT IS RELATED TO IT
UPDATE order_items
SET order_item_id = 60350
WHERE order_item_id = 5189;
```

5

```sql
--order_locations TABLE
WITH duplicates AS( --USE CTE TO DISPLAY DUPLICATE RECORDS --STEP 2
    SELECT * --VIEWING FOR DUPLICATE RECORDS IN order_locations TABLE--STEP 1
    FROM order_locations
    WHERE (city, country, region, state, zip_code, order_customer_id) IN (
        SELECT
            city, country, region, state, zip_code,order_customer_id
        FROM order_locations
        GROUP BY
            city, country, region, state, zip_code,order_customer_id
        HAVING COUNT(*) >1)
    ORDER BY city, country, location_id DESC)
--NOTE: ENSURE THAT THE COUNT OF THE DUPLICATE RECORDS IS EXACTLY 2
SELECT --SEPARATE ORIGINAL IDs TO DUPLICATE IDs --STEP 3
    MAX(CASE WHEN row_num % 2 <> 0 THEN location_id END) AS duplicated_id,
    MAX(CASE WHEN row_num % 2 = 0 THEN location_id END) AS original_id
FROM(
    SELECT location_id,
    ROW_NUMBER() OVER() AS row_num
    FROM duplicates
    ) AS numbered_data
GROUP BY CEIL(row_num / 2.0)
ORDER BY MIN(row_num);
```

duplicated_id integer	original_id integer
46563	44127
44405	42039
53143	51726
49787	48410
46086	45785
26537	20592
29812	29107
63523	62189
19216	15654
12812	11884
67913	67658
66510	63922
16366	14558
63848	13562

rows: 195 of 195 Query cor

6

```sql
--DELETE DUPLICATE RECORDS FROM order_locations TABLE
DELETE
FROM order_locations AS dup_id
USING order_locations AS dist_id
WHERE dup_id < dist_id
    AND dup_id.city = dist_id.city
    AND dup_id.country = dist_id.country
    AND dup_id.region = dist_id.region
    AND dup_id.state = dist_id.state
    AND dup_id.zip_code = dist_id.zip_code
    AND dup_id.order_customer_id = dist_id.order_customer_id
RETURNING dup_id.location_id AS deleted_location_id;
```

deleted_location_id integer
13562
16826
5955
14174
48193
1030
4628
10503
2970
1623

rows: 195 of 195 O

7

```sql
--UPDATE fact_table AS IT IS RELATED TO IT
UPDATE fact_table
SET order_item_id = dl.original_id
FROM deleted_location_id AS dl
WHERE fact_table.order_item_id = dl.duplicated_id;

--UPDATE order_items AS IT IS RELATED TO IT
UPDATE order_items
SET order_item_id = dl.original_id
FROM deleted_location_id AS dl
WHERE order_items.order_item_id = dl.duplicated_id;
```

order_items TABLE

1

```sql
--CHECKING DUPLICATE RECORDS IN order_items TABLE
SELECT order_date, item_discount_rate, item_qty, profit_per_order,
        status, order_item_cardprod_id, COUNT(*)
FROM order_items
GROUP BY order_date, item_discount_rate, item_qty, profit_per_order,
        status, order_item_cardprod_id
HAVING COUNT(*) >1; -- NO DUPLICATE RECORDS
```

shipments TABLE

1

```sql
--CHECKING DUPLICATE RECORDS IN shipments TABLE
SELECT type, days_for_shipping_real, days_for_shipping_sched, shipped_date,
        shipping_mode,delivery_status,late_delivery_risk, COUNT(*)
FROM shipments
GROUP BY type, days_for_shipping_real, days_for_shipping_sched, shipped_date,
        shipping_mode,delivery_status,late_delivery_risk
HAVING COUNT(*) > 1; -- NO DUPLICATE RECORDS
```

departments TABLE

1

```sql
--CHECKING DUPLICATE RECORDS IN departments TABLE
SELECT name, COUNT(*)
FROM departments
GROUP BY name
HAVING COUNT(*) >1; -- NO DUPLICATE RECORDS
```

stores TABLE

1

```sql
--CHECKING DUPLICATE RECORDS IN stores TABLE
SELECT latitude, longitude, market, department_id, COUNT(*)
FROM stores
GROUP BY latitude, longitude, market, department_id
HAVING COUNT(*) > 1; --NO DUPLICATE RECORDS
```

fact_table TABLE

1

```sql
--CHECKING DUPLCATE RECORDS IN fact_table TABLE
SELECT customer_id, store_id, product_id, order_item_id, shipment_id, COUNT(*)
FROM fact_table
GROUP BY customer_id, store_id, product_id, order_item_id, shipment_id
HAVING COUNT(*) > 1; --NO DUPLICATE RECORDS
```

DATASET RESOURCES:

OWNER: Shashwat Tiwari
YEAR: 2020
NAME: DataCo SMART Supply Chain for Big Data Analysis
VERSION 1
DATE RETRIEVED: January 16, 2024
FROM: https://www.kaggle.com/datasets/shashwatwork/dataco-smart-supply-chain-for-big-data-analysis

Acknowledgements

Data Engineering Pilipinas

University of the East – Caloocan, Google Developer Student Clubs

Department of Information and Communication Technology

FOR INQUIRIES, PLEASE CONTACT ME AT:

PHONE: 0945812260
EMAIL: cbmcamacho26@gmail.com
LINKED IN: https://www.linkedin.com/in/cris-bailon-camacho-04883a231