Neptune Duplicate Email Cleanup (email_address field)

This script is used to **find and clean duplicate email addresses** in our **Amazon Neptune** graph database.

The idea is simple — if multiple vertices have the same email, we'll keep the **latest record (based on lastModified)** and **remove the email_address** from the older ones.

It's written in Python and uses the **boto3 Neptune Data API**, so no Gremlin or external query console is required.

What the Script Does

Step 1: Setup

At the top, we configure:

- The **Neptune endpoint** and **region** (so boto3 knows which cluster to connect to).
- The **label** (like a table name, here Person) where our data lives.
- The **email field** (email_address) and **timestamp field** (lastModified) to detect which record is newer.
- The **DRY_RUN** flag — when this is True, it only reports what would be deleted but doesn't actually make any changes.

We also define a few **test email IDs** in a list called emails_to_check.

You can replace that list with real email IDs you want to scan.

Step 2: Fetch Records from Neptune

For each email in that list, the script runs a small **Cypher query** on Neptune to find any vertices that have that email.

It looks in both cases —

- if the email is stored as a **single value**, or
- if it's part of a **list** inside the vertex (some vertices may store multiple emails).

Example of what it runs:

```
MATCH (n:Email)
WHERE 'alpha@example.com' IN n.email_address OR n.email_address =
'alpha@example.com'
RETURN n.id AS id, n.email_address AS email_address, n.lastModified AS
lastModified
```

Then it collects all these results into a records list.

Step 3: Build a DataFrame

Once we've fetched all the matching vertices, we convert them into a **Pandas DataFrame** for easy processing.

Example of what it looks like:

~id	email_address	lastModified
a123	alpha@example.com	2025-10-30
a456	alpha@example.com	2025-11-02
b789	user2@company.com	2025-10-10

This gives us a nice tabular view of all duplicates we'll be cleaning.

Step 4: Handle Email Lists

Sometimes Neptune stores emails as lists — for example:

```
["alpha@example.com", "a.alt@example.com"]
```

To handle this, the script parses each email_address value and **flattens** all list elements into single rows.

After this step, each row represents one unique email tied to a vertex ID.

Step 5: Detect Duplicates

We now check for emails that appear in more than one vertex (duplicated() in Pandas).

If there are duplicates:

- We **sort them by lastModified (descending)** so the newest one comes first.
- Then we **mark that first one as "keeper"**.
- The rest are the **ones we'll nullify**.

Example:

email	vertex	lastModified	Action
alpha@example.com	a456	2025-11-02	Keep
alpha@example.com	a123	2025-10-30	Nullify

Step 6: Nullify Older Duplicates

If it's not a dry run, the script loops through each older vertex and runs a Cypher update like this:

```
MATCH (n:Person {id:'a123'}) REMOVE n.email_address
```

That basically removes the email property but keeps the vertex itself.

This way, the duplicate record isn't deleted — it's just stripped of the duplicate email.

The script does this in **batches** (100 records at a time) to avoid overloading Neptune.

Step 7: Print Summary

At the end, it prints a summary:

```
Duplicates found: 4 | To nullify: 3
Dry run mode -- no updates applied.
Total checked emails: 4
Dry run: True
```

So you know exactly what happened and whether anything was modified.

Why This Approach

- It’s **safe** — we’re not deleting nodes, only removing duplicate email values.
- It’s **controlled** — DRY_RUN lets you preview everything before applying.
- It’s **scoped** — only the emails you pass in the list are checked, nothing else.
- It’s **repeatable** — you can rerun this anytime without harming the graph.