Neptune Duplicate Email Cleanup (Keep Latest Record Only)

1. Objective

Remove duplicate email records from Amazon Neptune by keeping only the **latest vertex per email** (based on a timestamp property such as lastModified) and **nullifying** the email property for all older duplicates.

2. Pre-Requisites

Requirement	Details
AWS Access	IAM role with neptune-db:*, s3:*, and cloudwatch:*
Network	Job (Glue or SageMaker) must run in same VPC/subnet as Neptune
Libraries	boto3, gremlinpython, pandas, requests, tqdm
S3 Buckets	s3://data-cleanup-reports/ for reports
Fields Required	id, email, and lastModified (ISO 8601 or epoch)
Execution Mode	SageMaker notebook or Glue Python Shell

3. Cleanup Flow

1. Extract vertices with email and lastModified
2. Identify duplicate emails
3. Keep latest vertex for each email
4. Nullify email property on older duplicates
5. Generate cleanup report and upload to S3

4. Configuration

config/cleanup_neptune_email.json

```
{
  "endpoint_wss":
"wss://my-neptune.cluster-xyz.us-east-1.neptune.amazonaws.com:8182/greml
in",
  "label": "Person",
  "email_prop": "email",
  "timestamp_prop": "lastModified",
  "batch_size": 500,
  "report_bucket": "data-cleanup-reports",
  "report_prefix": "neptune_email/",
  "dry_run": true
}
```

5. Implementation

scripts/neptune_email_dedupe_dataframe.py

```
import json, boto3, pandas as pd, time
from datetime import datetime
from gremlin_python.driver import client, serializer

cfg = json.load(open('config/cleanup_neptune_email.json'))
g = client.Client(cfg['endpoint_wss'], 'g',
    message_serializer=serializer.GraphSONSerializersV3d0())

label = cfg['label']
email = cfg['email_prop']
ts = cfg['timestamp_prop']
dry_run = cfg['dry_run']
batch_size = cfg['batch_size']

# Step 1 -- Extract vertices
```

```
query = f"g.V().hasLabel('{label}').has('{email}').valueMap(true)"
records = g.submit(query).all().result()

def unwrap(row):
    out = {'id': row['id']}
    for k, v in row.items():
        if k == 'id': continue
        out[k] = v[0] if isinstance(v, list) and v else v
    return out

df = pd.DataFrame([unwrap(r) for r in records])
df = df.dropna(subset=[email])
print(f"Total vertices fetched: {len(df)}")

# Step 2 -- Find duplicates
dupes = df[df.duplicated(email, keep=False)]
if dupes.empty:
    print("No duplicate emails found.")
    exit()

# Step 3 -- Keep latest per email
dupes_sorted = dupes.sort_values(ts, ascending=False)
keepers = dupes_sorted.drop_duplicates(email, keep='first')
to_nullify = dupes_sorted[~dupes_sorted['id'].isin(keepers['id'])]

print(f"Duplicates found: {len(dupes)} | To nullify: {len(to_nullify)}")

# Step 4 -- Nullify older duplicates
if not dry_run:
    for i, vid in enumerate(to_nullify['id']):
        q = f"g.V('{vid}').properties('{email}').drop()"
        g.submit(q).all().result()
        if (i + 1) % batch_size == 0:
            time.sleep(0.3)
    print("Cleanup completed.")
else:
    print("Dry run mode -- no updates applied.")

# Step 5 -- Report
report = {
  "system": "Neptune",
  "label": label,
  "duplicates_found": int(len(dupes)),
  "nullified": int(len(to_nullify)),
  "retained": int(len(keepers)),
  "dry_run": dry_run,
```

```
  "timestamp": datetime.utcnow().isoformat()
}
boto3.client('s3').put_object(
  Bucket=cfg['report_bucket'],

Key=f"{cfg['report_prefix']}email_dedupe_{datetime.utcnow().strftime('%Y
%m%d_%H%M%S')}.json",
  Body=json.dumps(report)
)
print("Report written to S3.")
```

---

6. Execution Steps

1. Upload config and script to your environment:

```
infra/data_cleanup/config/cleanup_neptune_email.json
infra/data_cleanup/scripts/neptune_email_dedupe_dataframe.py
```

1. Install dependencies:

```
pip install -r requirements.txt
```

2. Run the script:

```
python scripts/neptune_email_dedupe_dataframe.py
```

3. Check the output in the console.
4. Review the report in:

```
s3://data-cleanup-reports/neptune_email/
```

---

7. Verification

Gremlin Queries

```
// Remaining duplicates (should return none)
g.V().has('email').groupCount().by('email')
   .unfold().where(select(values).is(gt(1)))

// Spot check a specific email
g.V().has('email','test@example.com').valueMap(true)
```