

Neo4j Sandbox 구축

단계1: Neo4j Sandbox 접속

Log in

Log in to Neo4j to continue

Log in

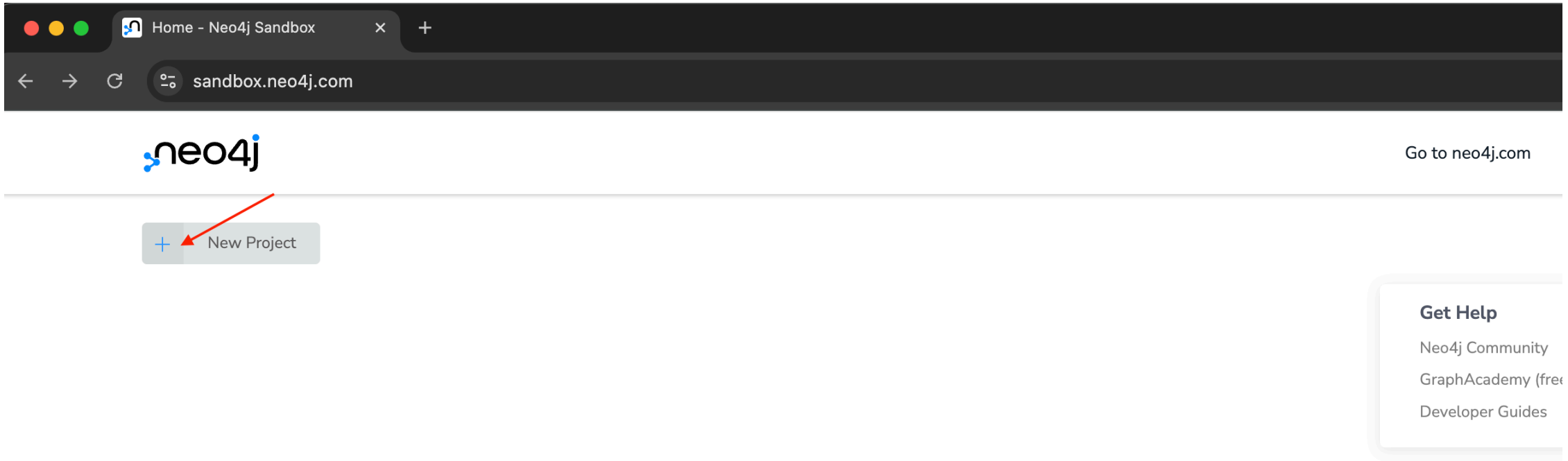
Don't have an account? [Sign up](#)

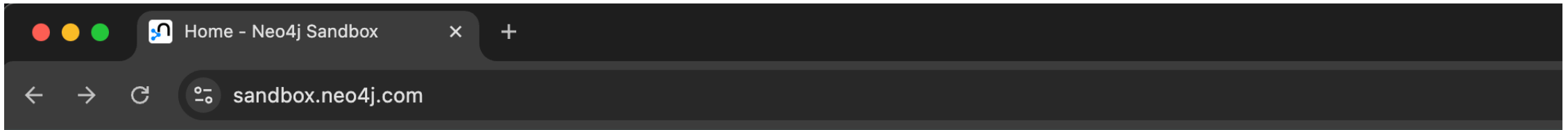
OR



Continue with Google

단계2: New Project





Your own data



Blank Sandbox

A sandbox to explore connections in your own data
- by importing CSV, using Neo4j drivers or any other
way you like.



For Data Scientists

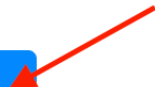


Blank Sandbox - Graph Data Science

Leverage Neo4j Graph Data Science library to
explore graph algorithms for analytics and feature
engineering.

Project : **Blank Sandbox**

Create and Download credentials



단계3: 생성된 DB의 접속 정보 확인

The screenshot displays the Neo4j Sandbox interface. At the top, a table lists the sandbox status:

Name	Status
Blank Sandbox	Running Expires in about 3 days

Below the table, there is a row of tabs: Actions, **Connection details** (selected), Connect via drivers, Backups, AuraDB Upload, and Connect via HTTP. A red arrow points to the 'Open' button in the top right corner of the table row. Another red arrow points to the 'Connection details' tab.

The 'Connection details' section contains the following information:

- Username: neo4j
- Password: workbook-temperature-challenges
- IP Address: 3.219.247.27
- HTTP Port: 7474
- Bolt Port: 7687

Below this, the Bolt URL is shown as: bolt://3.219.247.27:7687

The Websocket Bolt URL is highlighted with a red box and is: bolt+s://a8bcf0e4e76ac94ef51633bde51a8fa1.neo4jsandbox.com:7687

Finally, the Websocket Bolt URL (Port 443) is shown as: bolt+s://a8bcf0e4e76ac94ef51633bde51a8fa1.bolt.neo4jsandbox.com:443

데이터셋 임포트



Connect to instance

Scheme

bolt+s



Connection URL

a8bcf0e4e76ac94ef51633bde5



Connect with SSO

Database user

neo4j

Password

.....


Connect

Cancel

단계1: data-importer 접속

- 생성된 DB의 접속 정보 사용

단계2: 파일 업로드



Drag & Drop
or [browse](#)

Supports CSV, TSV

🔍 Add node

🗑️

Label ⓘ

Node name

즐거찾기

AirDrop

🕒 최근 항목

🚀 응용 프로그램

📁 데스크탑

📄 문서

📶 다운로드

iCloud

📁 iCloud Drive

📁 공유

위치

< > 다운로드

⋮ ⏏

📄

📁

🔍

이름	크기	종류	추가된 날짜
Data Preprocessing.ipynb	336KB	문서	오늘 오후 4:03
movie_meta.csv	489KB	CSV 문서	오늘 오후 4:03
movie_rating.csv	855KB	CSV 문서	오늘 오후 4:03
movie_keyword.csv	253KB	CSV 문서	오늘 오후 4:03
movie_genre.csv	49KB	CSV 문서	오늘 오후 4:02
Spectacle	4.4MB	응용 프로그램	2022년 1월 27일 오전 8:

Mapping

from file

+

ined

단계3: movied 노드 생성

Files

Browse

...

^ movie_genre.csv

...

^ movie_keyword.csv

...

^ movie_rating.csv

...

^ movie_meta.csv

...

● movied

2086

● title

Nick of Time

● overview

Gene Watson ...

👁

▶ Run import

...

🔍

🗑

movied

📄

🔍

Definition

Constraints & Indexes (2)

Label ⓘ

Name

movied

File ⓘ

Name

movie_meta.csv

Filter file

Properties

Map from file

+

<input type="checkbox"/>	Name	Type	Column	ID
<input type="checkbox"/>	movied ✎	integer ▼	movied ▼	🔑
<input type="checkbox"/>	title ✎	string ▼	title ▼	🔑
<input type="checkbox"/>	overview ✎	string ▼	overview ▼	🔑

단계4: rating 노드 생성

Files

Browse ...

^ movie_genre.csv ...

^ movie_keyword.csv ...

^ movie_meta.csv ...

▼ movie_rating.csv ...

- id 0
- userId 1
- movied 1371
- rating 2.832339524...

Show results

Run import

movied

rating

Definition Constraints & Indexes (2)

Name

rating

File ⓘ

Name

movie_rating.csv

Filter file

Properties

Map from file

<input type="checkbox"/>	Name	Type	Column	ID
<input type="checkbox"/>	id	integer	id	<input checked="" type="checkbox"/>
<input type="checkbox"/>	userId	integer	userId	<input type="checkbox"/>
<input type="checkbox"/>	movied	integer	movied	<input type="checkbox"/>
<input type="checkbox"/>	rating	float	rating	<input type="checkbox"/>

단계5: userId 노드 생성

Files

Browse

...

^ movie_genre.csv

...

^ movie_keyword.csv

...

^ movie_meta.csv

...

▼ movie_rating.csv

...

● id

0

● userId

1

● movieId

1371

● rating

2.832339524...

Show results

Run import

...

rating

userId

Definition

Constraints & Indexes (2)

Label ⓘ

Name

userId

File ⓘ

Name

movie_rating.csv

Filter file

Properties

Map from file

+

☐

Name

☐

userId

Type

integer

▼

Column

userId

▼

ID

key

단계6: genre 노드 생성

Files

Browse ...

movie_genre.csv ...

- movieId 2086
- genre Crime

movie_keyword.csv ...

movie_meta.csv ...

movie_rating.csv ...

Show results

Run import

rating

userId

genre

Definition Constraints & Indexes (2)

Label ⓘ

Name

genre

File ⓘ

Name

movie_genre.csv

Filter file

Properties

Map from file +

<input type="checkbox"/>	Name	Type	Column	ID
<input type="checkbox"/>	movieId	integer	movieId	
<input type="checkbox"/>	genre	string	genre	

단계7: keyword 노드 생성

Files

Browse ...

movie_genre.csv ...

movie_keyword.csv ...

- movieId 2086
- keyword assassination

movie_meta.csv ...

movie_rating.csv ...

Show results

Run import

rating

userId

keyword

genre

Definition Constraints & Indexes (2)

Label ⓘ

Name

keyword

File ⓘ

Name

movie_keyword.csv

Filter file

Properties

Map from file +

<input type="checkbox"/>	Name	Type	Column	ID
<input type="checkbox"/>	movieId	integer	movieId	
<input type="checkbox"/>	keyword	string	keyword	


관계성 생성

단계1: Neo4j Sandbox 접속



[Go to neo4j.com](https://neo4j.com)

+ New Project

Name	Status
 Blank Sandbox	Running Expires in about 3 days

Open

Actions

Connection details

Connect via drivers

Backups

AuraDB Upload

Connect via HTTP

Username: neo4j

IP Address: 3.219.247.27

Password: workbook-temperature-challenges

HTTP Port: 7474

Bolt Port: 7687

Neo4j Bloom

[Bloom Video Series](#)
[Bloom Guide](#)

Get Help

[Neo4j Community](#)
[GraphAcademy \(free\)](#)
[Developer Guides](#)

단계2: Nodes 확인

The screenshot displays the Neo4j Desktop interface. On the left, the 'Database Information' sidebar shows the 'neo4j' database selected. Under 'Node labels', a red box highlights the labels: `*(39,961)`, `genre`, `keyword`, `movied`, `rating`, and `userId`. The 'Relationship types' section indicates 'No relationships in database'. The 'Property keys' section lists `genre`, `id`, `keyword`, and `movied`.

The main workspace shows a Cypher query in the top bar: `neo4j$ MATCH (n:genre) RETURN n LIMIT 25`. Below the query, a graph visualization displays 20 orange circular nodes representing movie genres. The nodes are labeled with various genres: `Adventure`, `Horror`, `Family`, `Comedy`, `Music`, `Action`, `Romance`, `Mystery`, `War`, `Animation`, `Crime`, `Documentary`, `Fantasy`, `Science Fiction`, `Thriller`, `Drama`, `Biography`, `History`, `War`, and `Animation`.

On the right, the 'Overview' panel shows 'Node labels' with `*(20)` and `genre (20)`. Below this, it states 'Displaying 20 nodes, 0 relationships.'

단계3: RATED 생성

```
MATCH (r:rating), (u:userId {userId: r.userId}), (m:movieId {movieId: r.movieId})  
MERGE (u)-[:RATED {score: r.rating}]->(m)
```

The screenshot displays the Neo4j Desktop interface. On the left, the 'Database Information' sidebar shows the database 'neo4j' selected. Under 'Node labels', there are buttons for '(39,961) genre', 'keyword', 'movieId', 'rating', and 'userId'. Under 'Relationship types', there is a button for '(31,920) RATED', which is highlighted by a red arrow. The main workspace shows a Cypher query editor with the same query as above. A red arrow points from the 'RATED' relationship type in the sidebar to the query. Below the query editor, a message states: 'Set 31920 properties, created 31920 relationships, completed after 7786 ms.' The interface also includes a 'Table' view icon, a 'Warn' icon, and a 'Code' icon.

```
MATCH p=()-[r:RATED]->>() RETURN p LIMIT 25
```

neo4j\$ MATCH p=()-[r:RATED]->>() RETURN p LIMIT 25

Graph

Table

Text

Code

Overview

Node labels

- * (27)
- userId (2)
- movieId (25)

Relationship types

- * (25)
- RATED (25)

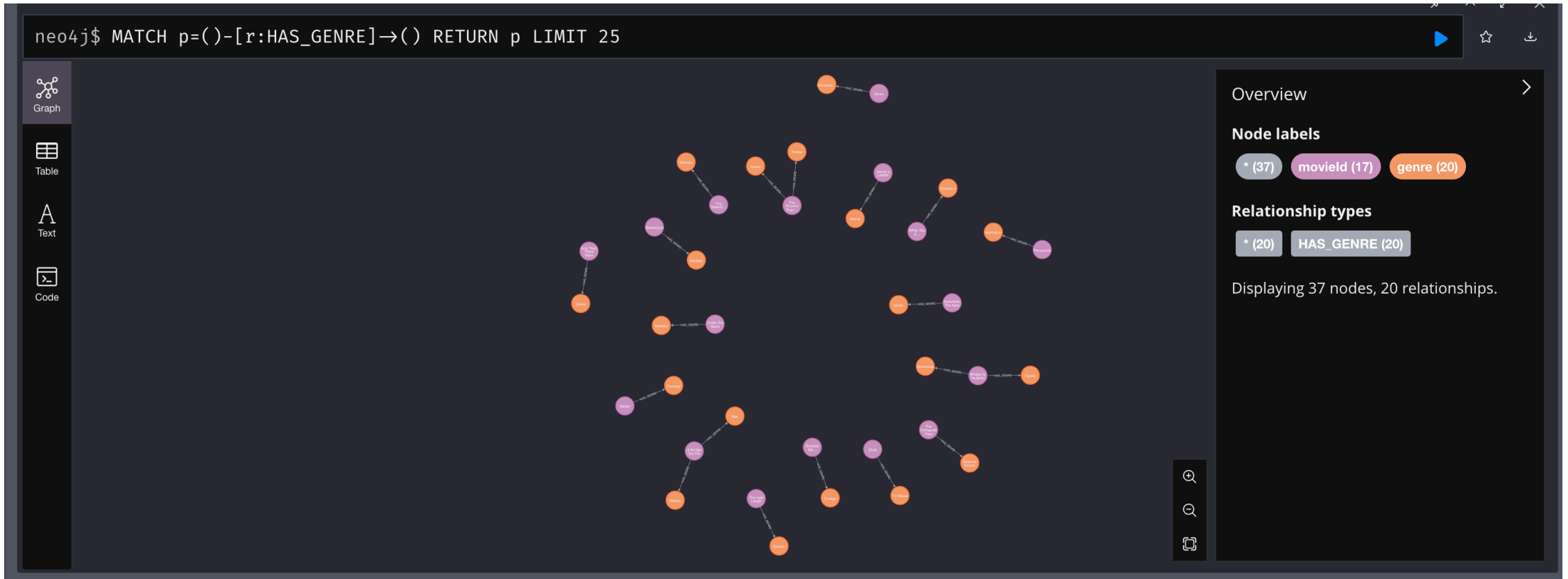
Displaying 27 nodes, 25 relationships.

단계4: HAS_GENRE 생성

```
MATCH (m:movieId), (g:genre)
WHERE m.movieId = g.movieId
MERGE (m)-[:HAS_GENRE]->(g)
```

The screenshot displays the Neo4j Desktop interface. On the left, the 'Database Information' panel shows the database 'neo4j' selected. Under 'Node labels', there are labels for 'movieId' (39,961 instances), 'genre' (39,961 instances), 'keyword', 'rating', and 'userId'. Under 'Relationship types', there are 'HAS_GENRE' (31,940 instances) and 'RATED'. A red arrow points to the 'HAS_GENRE' label. The main panel shows a Cypher query being executed: `neo4j$ MATCH (m:movieId), (g:genre) WHERE m.movieId = g.movieId MERGE (m)-[:HAS_GENRE]->(g)`. The query was successful, creating 20 relationships in 14 ms. The left sidebar contains icons for 'Table', 'Warn', and 'Code'.

```
MATCH p=()-[r:HAS_GENRE]->>() RETURN p LIMIT 25
```



단계5: HAS_KEYWORD 생성

```
MATCH (m:movieId), (k:keyword)
WHERE m.movieId = k.movieId
MERGE (m)-[:HAS_KEYWORD]->(k)
```

The screenshot displays the Neo4j Desktop interface. On the left, the 'Database Information' panel shows the database 'neo4j' selected. Under 'Node labels', there are labels for 'genre', 'keyword', 'movieId', 'rating', and 'userId'. Under 'Relationship types', there are 'HAS_GENRE', 'HAS_KEYWORD', and 'RATED'. A red arrow points to the 'HAS_KEYWORD' relationship type. On the right, the main workspace shows a Cypher query being executed: `MATCH (m:movieId), (k:keyword) WHERE m.movieId = k.movieId MERGE (m)-[:HAS_KEYWORD]->(k)`. The query is highlighted with a red arrow. Below the query, a message states: 'Created 5832 relationships, completed after 658 ms.' The interface also includes a sidebar with icons for 'Table', 'Warn', and 'Code'.

```
MATCH p=()-[r:HAS_KEYWORD]->() RETURN p LIMIT 25
```

