



MODUL PRAKTIKUM

SD3203-Teknologi Basis Data

**Program Studi Sains Data
Fakultas Sains
Institut Teknologi Sumatera**

2025

MODUL 8

Graph Database

Graph Database

1. Tujuan Praktikum

- a. Memahami konsep dasar Graph Database dan struktur datanya (Node, Relationship, Property, Path).
- b. Mampu membuat dan memodifikasi data dalam bentuk graf menggunakan Cypher.

2. Konsep Dasar

Basis data graph (graph database) adalah jenis basis data yang menggunakan struktur graph untuk menyimpan data. Salah satu aplikasi yang dapat diterapkan dalam graph database adalah Neo4j. Neo4J adalah sistem manajemen basis data graph yang dikembangkan oleh Neo4J, Inc., sebuah perusahaan yang berbasis di San Francisco Bay Area, Amerika Serikat. Hasil perhitungan peringkat DB-Engines menunjukkan Neo4J menempati peringkat pertama sistem basis data graph paling populer mengalahkan Microsoft Azure Cosmos DB, Arango DB, Orient DB, dan beberapa sistem manajemen basis data graph lain. Neo4J menggunakan bahasa query bernama Cypher. Cypher adalah sebuah bahasa deklaratif yang terinspirasi dari SQL untuk mendeskripsikan pola graph. Cypher pada awalnya dikembangkan eksklusif untuk Neo4J tapi kemudian dengan dirilisnya open Cypher kini Cypher bisa diterapkan pada SAP HANA dan Agens Graph. Cypher Query mendefinisikan node relationship dengan syntax CREATE dan MATCH. Neo4J seperti database graph lainnya bisa digunakan, dalam mempermudah, penyimpanan dan pengambilan data dalam bentuk graph. Misalnya adalah media sosial, system rekomendasi, fraud detection dsb-nya.

Pada Neo4J data direpresentasikan dalam bentuk Node dan Relasi. Node merupakan titik. Relasi adalah penghubung antar dua node. Baik Node dan Relasi memiliki property yang bisa kita sesuaikan. Sebelum menciptakan node atau Relasi, bisa kita definisikan atribut yang ada di tiap node atau Relasi tersebut. Hal ini opsional tidak wajib tapi merupakan best practice untuk dilakukan. Perancangan database di Neo4J hanya sebatas desain node dan relasi. Tidak ada istilah tabel atau database dalam Neo4J. Hal ini berbeda dengan basis data RDBMS

atau NoSQL lainnya. Untuk Neo4J, dalam satu database server Neo4J hanya ada satu database, kita tidak perlu mendefinisikan nama database, juga tidak ada table didalam database tersebut. Sehingga ketika kita ingin mengubah database maka seluruh file di direktori data diNeo4J harus dihapus secara manual. Cara ini yang paling cepat dibanding menghapus node dan relasi. Neo4j telah dikembangkan sejak tahun 2003, dapat ditulis dengan Bahasa Java, JRuby, Scala, Python, Clojure dan sebagainya.

Karakteristik tipe data dari neo4j adalah:

1. Bersifat optional, tidak memiliki struktur data, mudah saat melakukan perubahan skema.
2. Mudah untuk dimodelkan pada domain data sets yang umum seperti pada permodelan CMS (CompactMotion Switch) sebagai access controllist, database objek, Triple stores dll.
3. Penggunaannya pada semantic webdan EDF (External Data Format), linked Data, GIS (Geographic Information System), situs jejaring sosial dan lain sebagainya.

Bagian penting pembentuk graph database dalam Neo4j terdiri dari 6 bagian yaitu, Node, Relationship, Properties, Path, Traversal dan Query



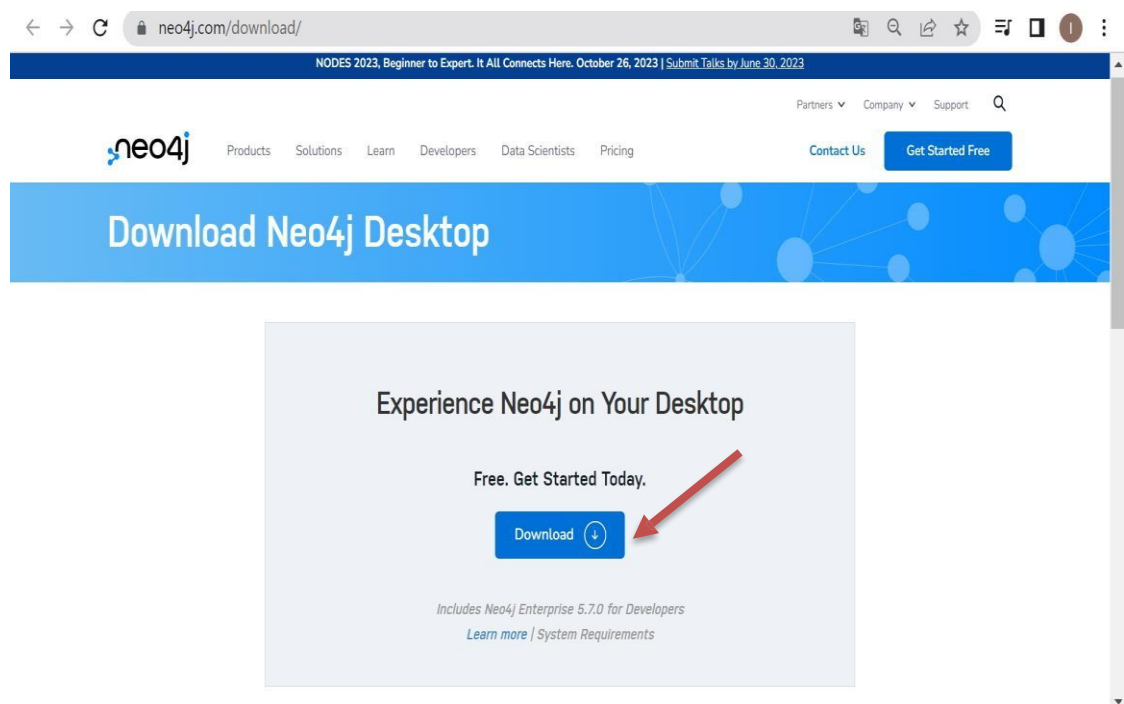
- Node adalah bagian yang paling mendasar yang membentuk sebuah graph. Dalam neo4j node dapat memiliki relationship dan property.
- Relationship merupakan salah satu kunci pembentuk graph database. Dengan adanya relationship memungkinkan untuk mencari data atau informasi yang berkaitan. Seperti halnya node, relationship juga dapat memiliki properti.
- Properti merupakan key pada setiap node yang umumnya adalah sebuah tipe data string. Dan nilai atau isi properti pada setiap key yang bertipe string dapat mencakup semua tipe data dan memungkinkan untuk fungsi array.

- Path merupakan node yang memiliki hubungan (relationship) disebut path. Misalnya hubungan antara node 1 dengan node 2, biasanya path berasal dari pengambilan secara query maupun hasil dari kunjungan traversal.
- Traversal memiliki arti mengunjungi setiap node, mengikuti setiap hubungan satu node ke node yang lain yang tentu saja berdasarkan aturan yang ada. Dalam banyak kasus, tiap node yang dikunjungi merupakan subgraph.
- Query merupakan kemampuan untuk menampilkan suatu data dari database. Bahasa query yang dipakai oleh Neo4j sekarang mendukung kemampuan untuk mengambil seluruh jalur terpendek, mengekstrak dengan mudah nilai pertama dari list dengan null menggunakan COALESCE, alias column dan menangani hubungan panjang variable.

3. Latihan Praktikum

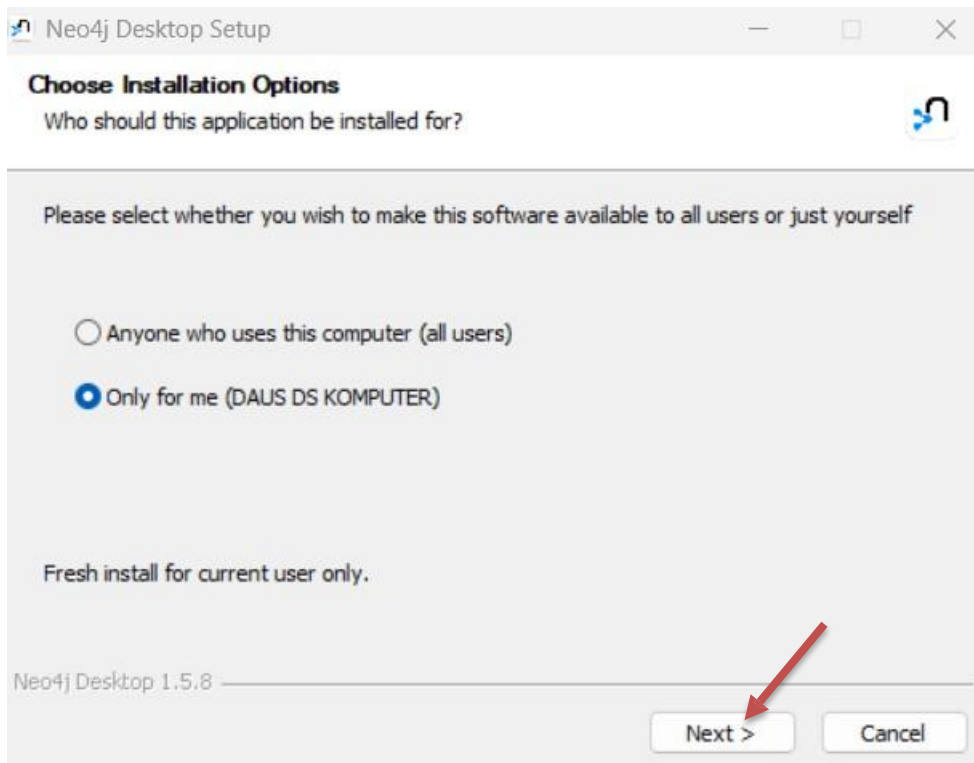
I. Cara Menginstall Aplikasi Neo4j

1. Download aplikasinya terlebih dahulu di <https://neo4j.com/download/>

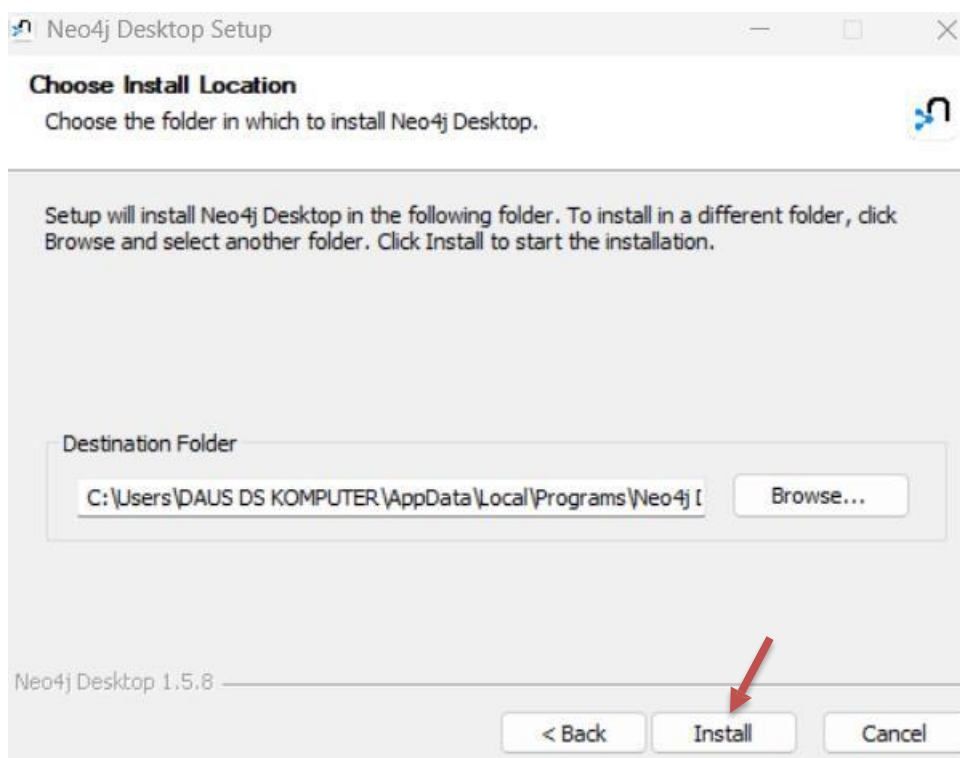


Kemudian klik download dan isilah data pribadi seperti dibawah ini

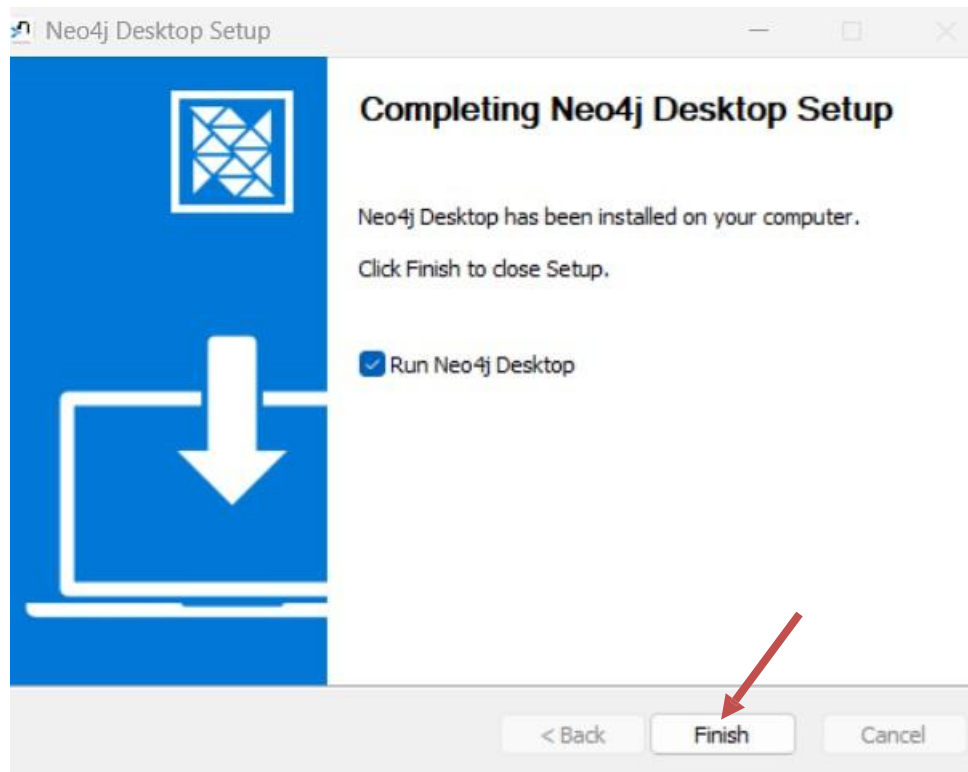
Get Started Free



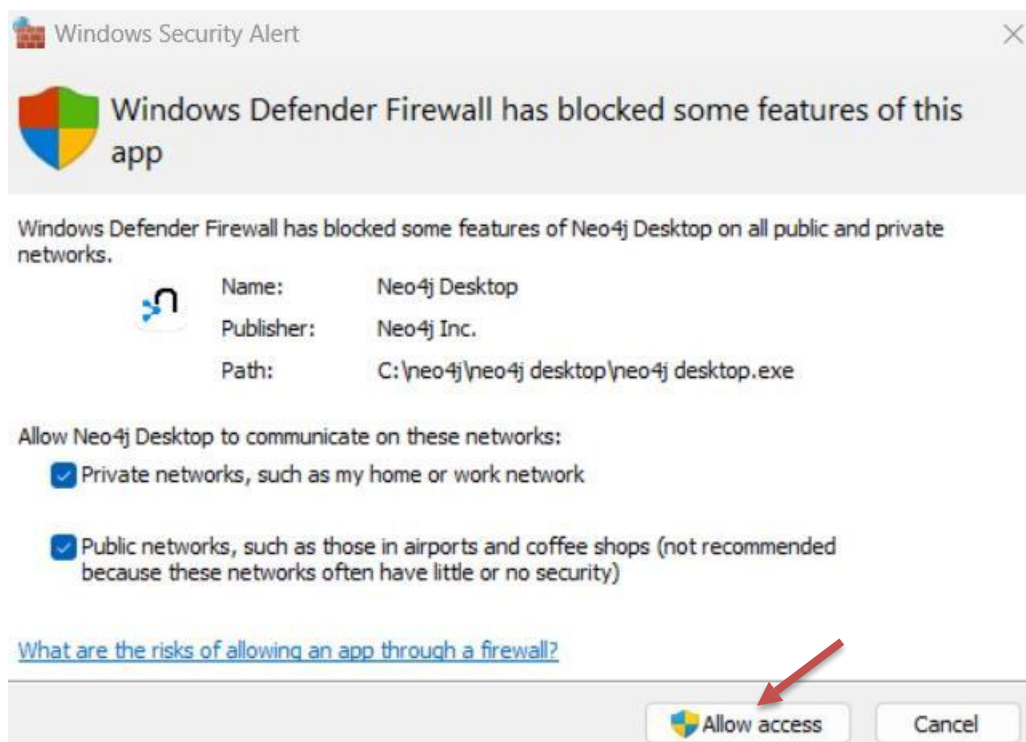
3. Pilih lokasinya. Lalu klik install



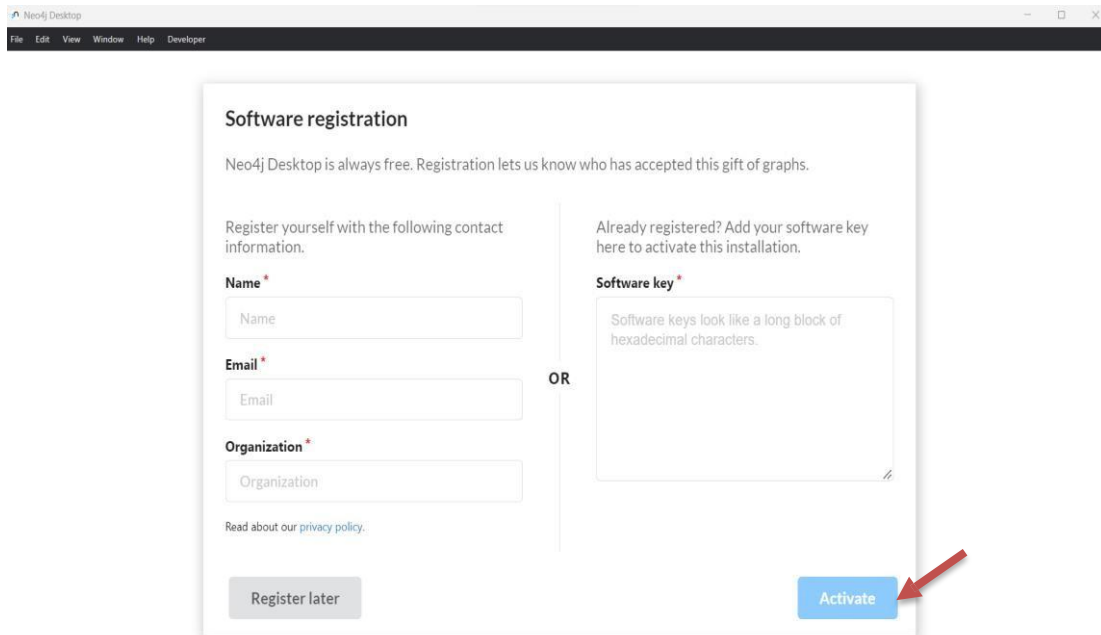
4. Klik finish aplikasi bisa digunakan



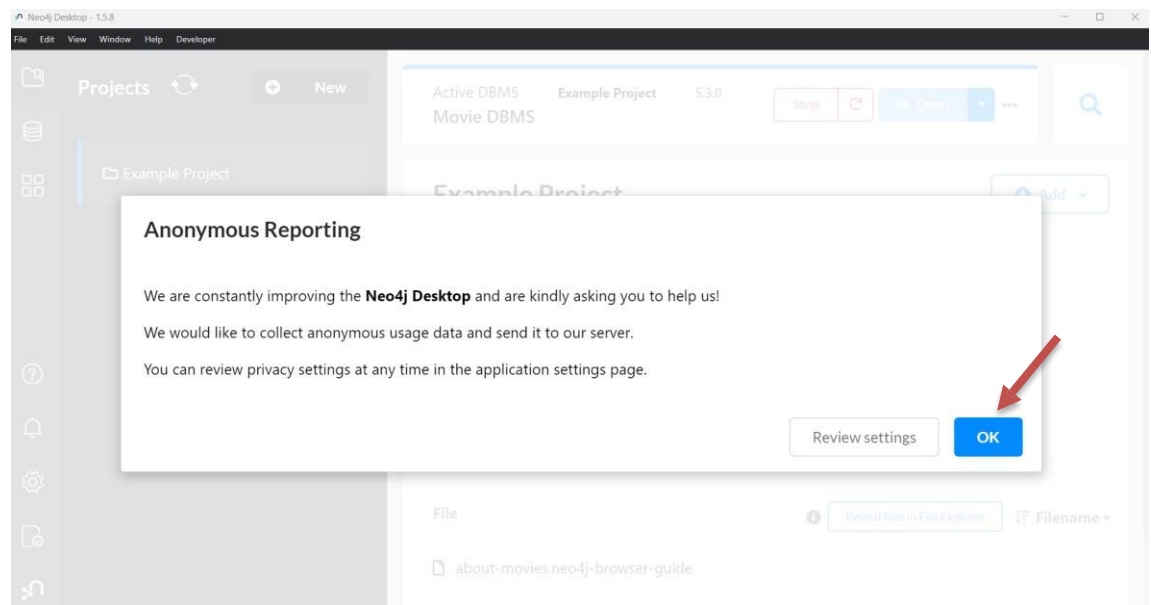
5. Selanjutnya run neo4j secara langsung dengan pilih centang 2 kali dan klik allow access. Kemudian pilih I Agree dan pastikan lokasi aplikasi tersebut sama seperti lokasinya saat di download komputer masing-masing.



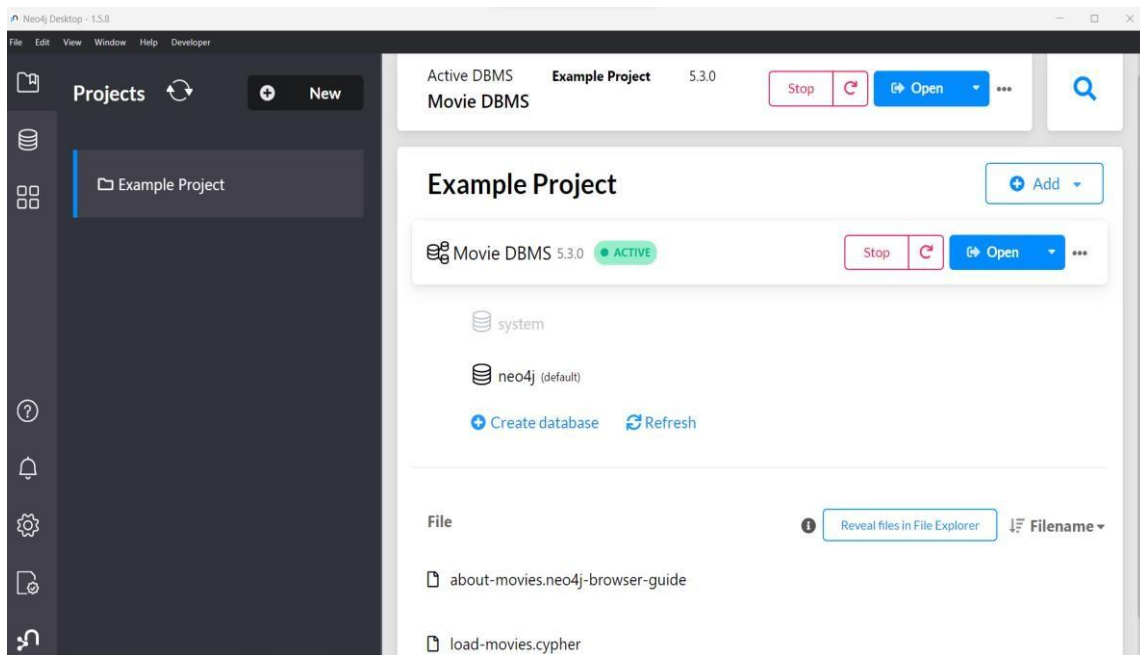
6. Langkah selanjutnya adalah isi data pribadi seperti langkah no.1 dan masukkan activation key yang telah disimpan di notepad++ dibagian software key lalu klik activate



Kemudian klik OK pada anonymous reporting

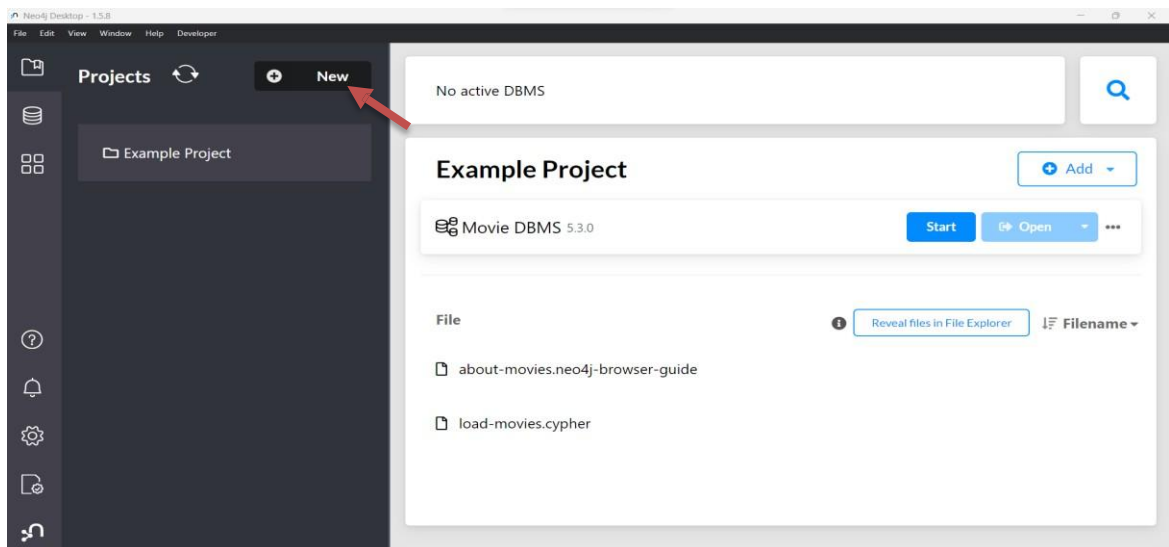


7. Jika sudah berhasil install aplikasi Neo4j maka tampilan layer akan seperti dibawah ini

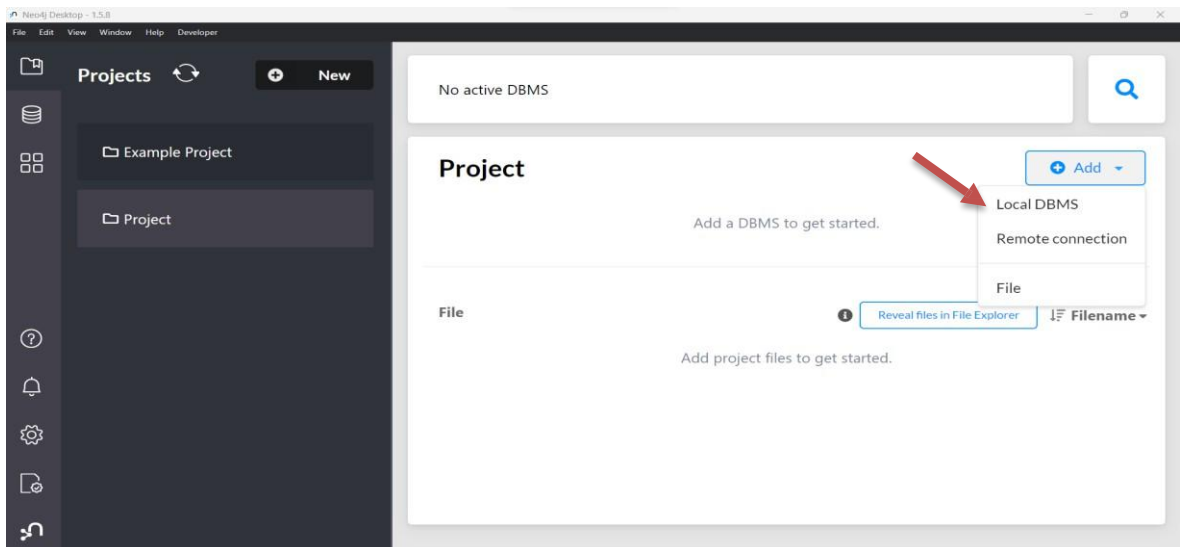


II. Cara Memulai Aplikasi Neo4j Desktop dan Membangun Graph Database

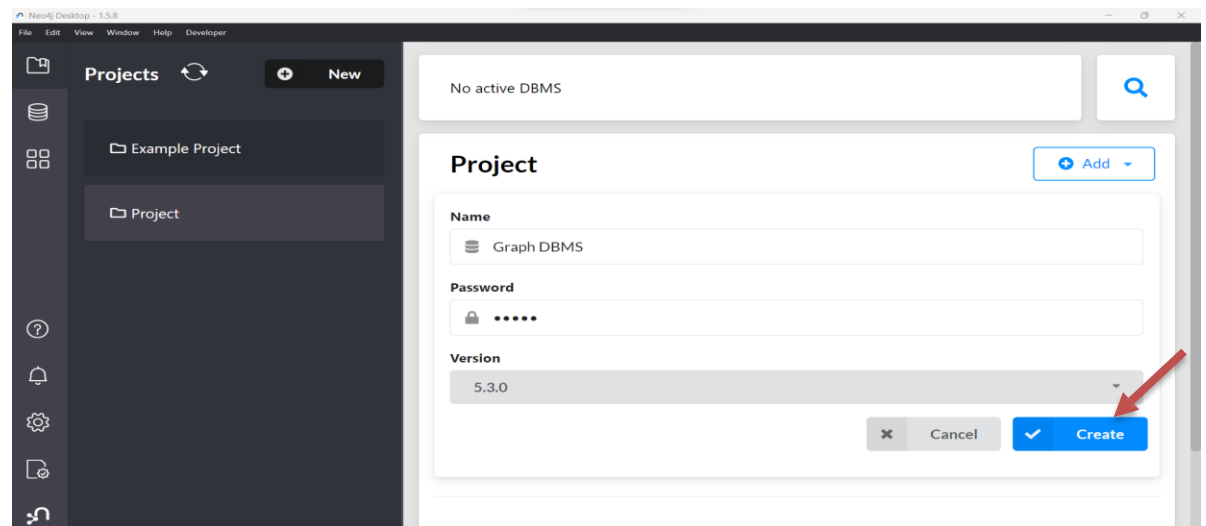
1. Bukalah aplikasi neo4j dan tampilan layar akan mnucul, lalu untuk membuat project baru dengan mengklik tombol “New” di bagian kiri atas layar kemudain pilih create project



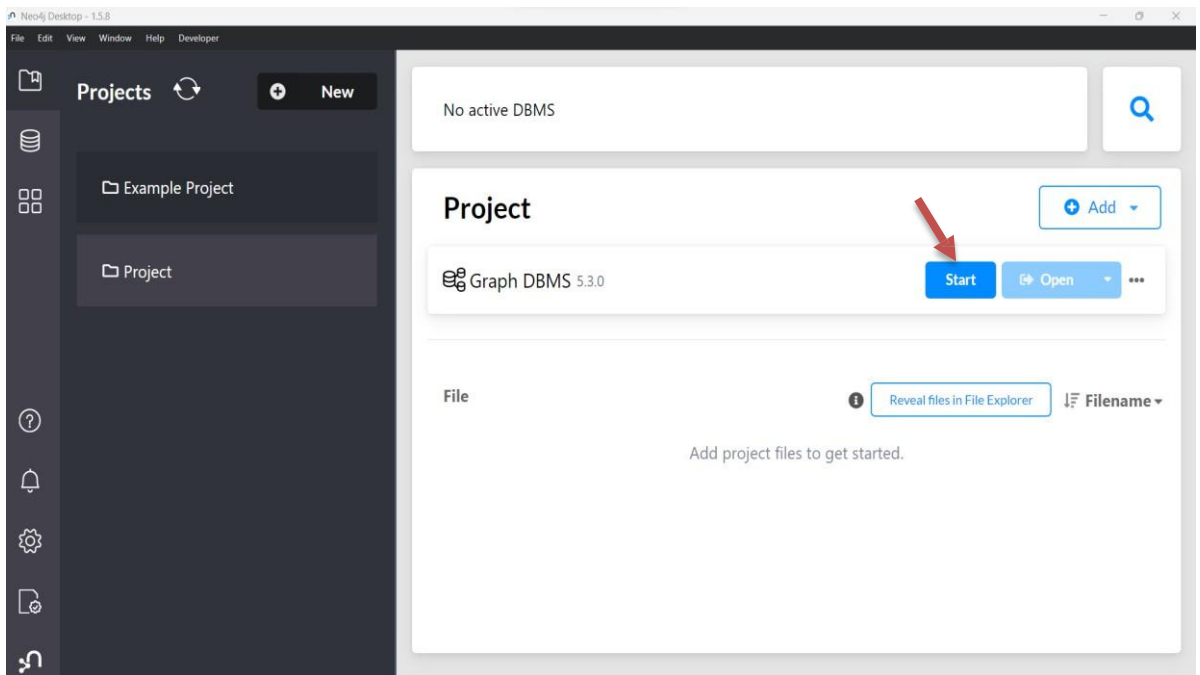
2. Tampilan Neo4j untuk new project seperti dibawah ini. Kemudian klik add lalu pilih local DBMS



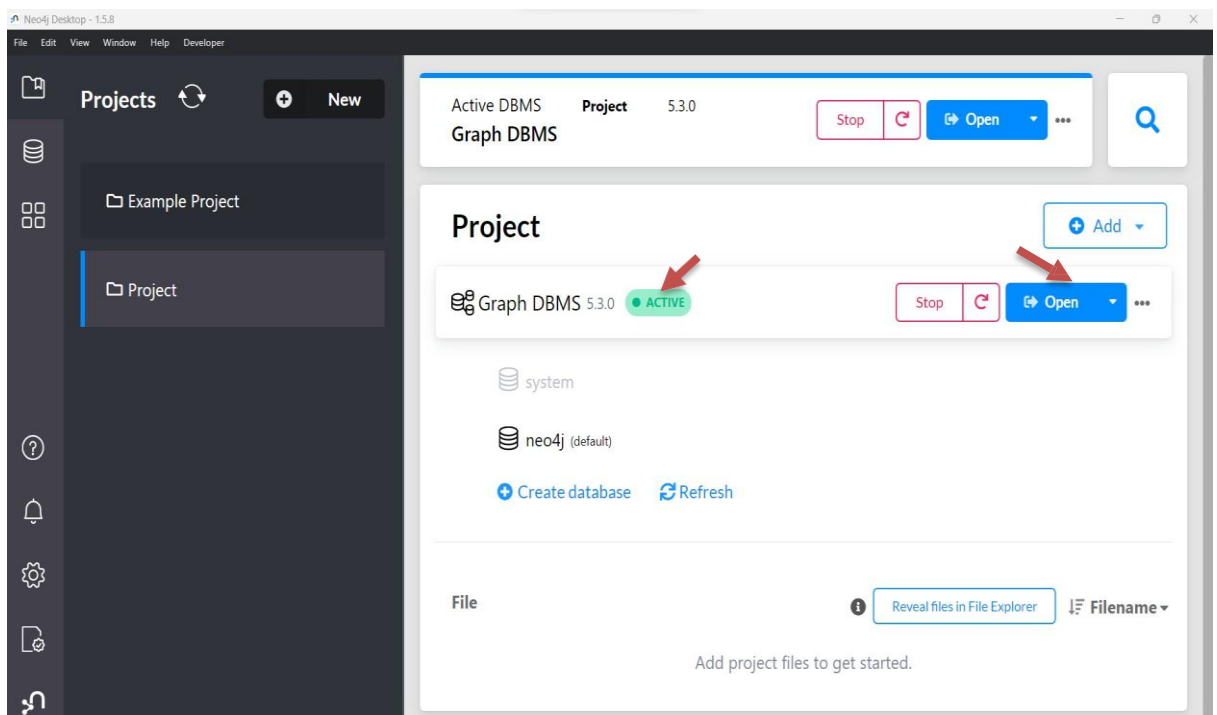
3. Langkah selanjutnya membuat nama database "Graph DBMS" lalu isi password minimal 8 karakter selanjutnya klik create. Password sesuai keinginan masing-masing.



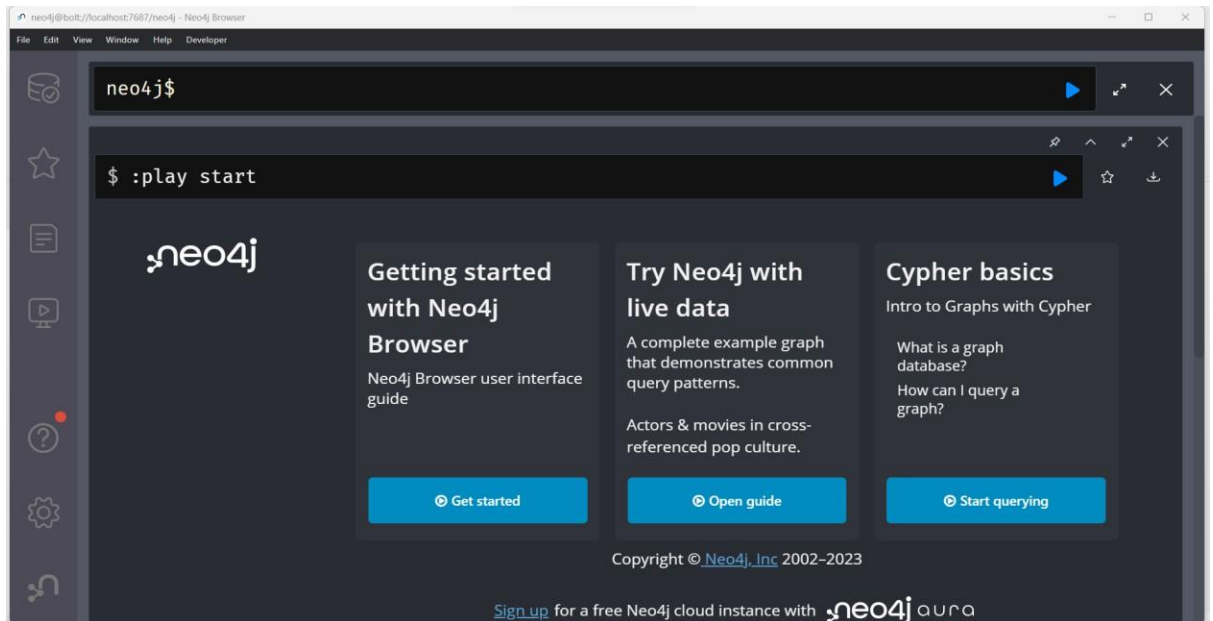
4. Kemudian untuk memulai Graph Database dengan klik start.



5. Tahap selanjutnya pastikan Graph DBMS sudah active kemudian pilih open with Neo4j browser.



Neo4j browser akan ditampilkan seperti dibawah ini

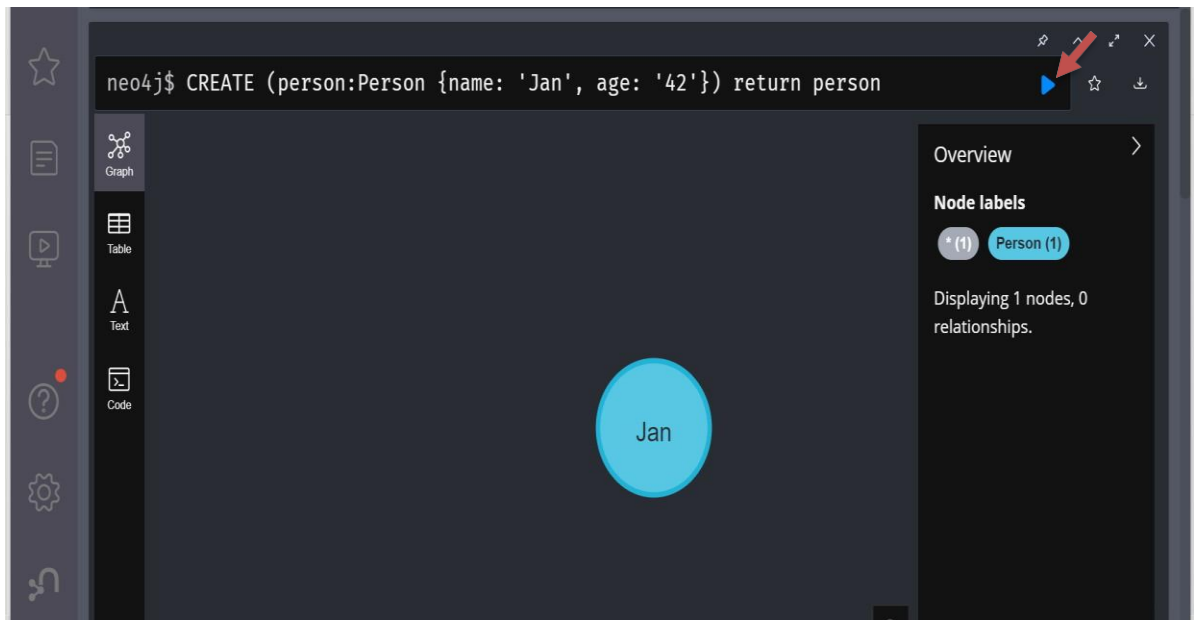


6. Tahap selanjutnya membuat node terlebih dahulu seperti dibawah ini

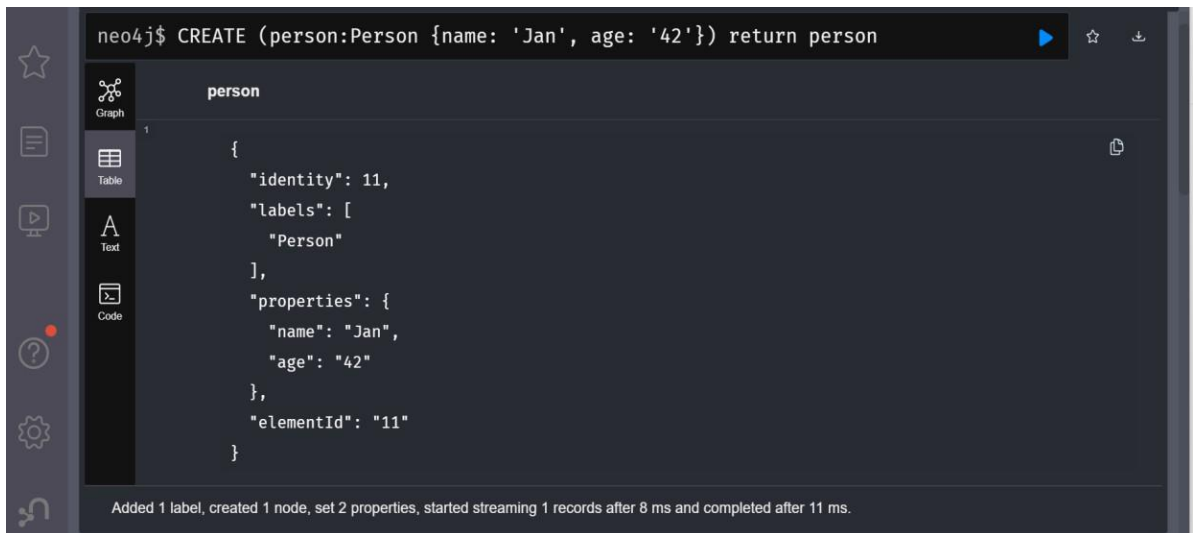
➤ **Membuat node tanpa relationships secara manual**

```
CREATE (person:Person {name: 'Jan', age: '42'}) return person
```

buatlah syntax seperti diatas lalu klik tanda panah biru



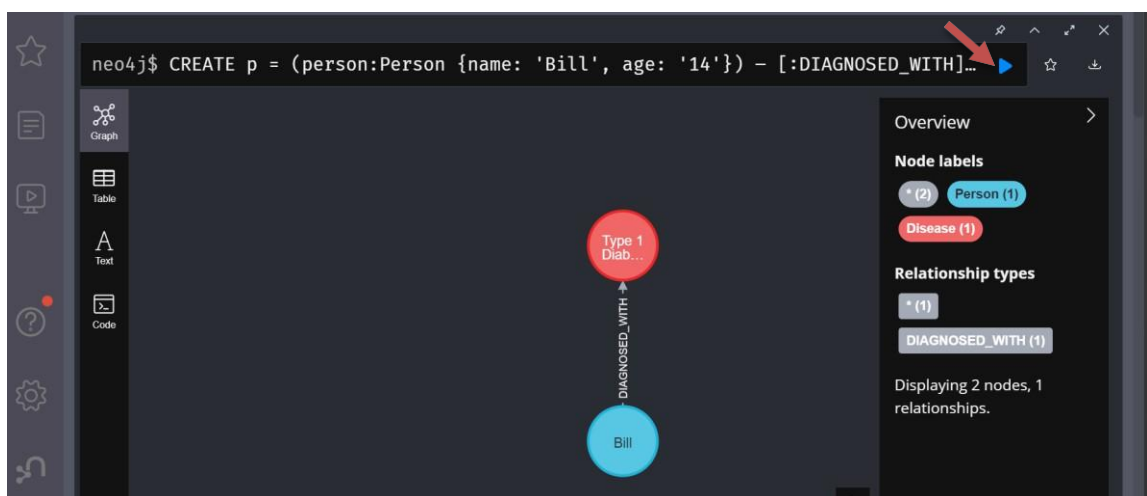
Jika sudah berhasil maka akan ditampilkan seperti ini dengan keterangan added 1 label, created 1 node



➤ Membuat node dengan relationships secara manual

```
CREATE p = (person:Person {name: 'Bill', age: '14'}) -
[:DIAGNOSED_WITH] → (disease:Disease { name: 'Type 1
Diabetes' }) RETURN p
```

buatlah syntax seperti diatas lalu klik tanda panah biru



Jika sudah berhasil maka akan ditampilkan seperti ini dengan keterangan added 2 label, created 2 node, set 3 properties dan created 1 relationship.

```
neo4j$ CREATE p = (person:Person {name: 'Bill', age: '14'}) -[:DIAGNOSED_WITH] -> (disease:Disease { name...
```

```
{
  "start": {
    "identity": 19,
    "labels": [
      "Person"
    ],
    "properties": {
      "name": "Bill",
      "age": "14"
    },
    "elementId": "19"
  },
  "end": {
    "identity": 20,
    "labels": [
      "Disease"
    ]
  }
}
```

Added 2 labels, created 2 nodes, set 3 properties, created 1 relationship, started streaming 1 records in less than 1 ms and completed after 6 ms.

7. Kemudian step-step selanjutnya dengan **membuat relationships** dengan menambahkan relationships diantara node. Berikut syntax ditampilkan dibawah ini.

```
MATCH (p:Person {name:'Jan'}), (f:Person {name:'Samantha'})
CREATE (p)-[:FRIENDS_WITH {since: 2009}]->(f)
```

buatlah syntax seperti diatas lalu klik tanda panah biru. Jika sudah berhasil maka akan ditampilkan seperti ini

```
neo4j$ MATCH (p:Person {name:'Jan'}), (f:Person {name:'Samantha'}) CREATE (p)-[:FRIENDS_WITH {since: 2009}]->(f)
```

```
(no changes, no records)
```

Completed after 33 ms.

8. Kita dapat memperbarui atau **update node properties** dengan menggunakan syntax dibawah ini

```
MATCH (person:Person { name: 'Jan' }) SET person.profession = 'Software Engineer' RETURN person
```

buatlah syntax seperti diatas lalu klik tanda panah biru.

```
neo4j$ MATCH (person:Person { name: 'Jan' }) SET person.profession = 'Software Engineer' RETURN person
```

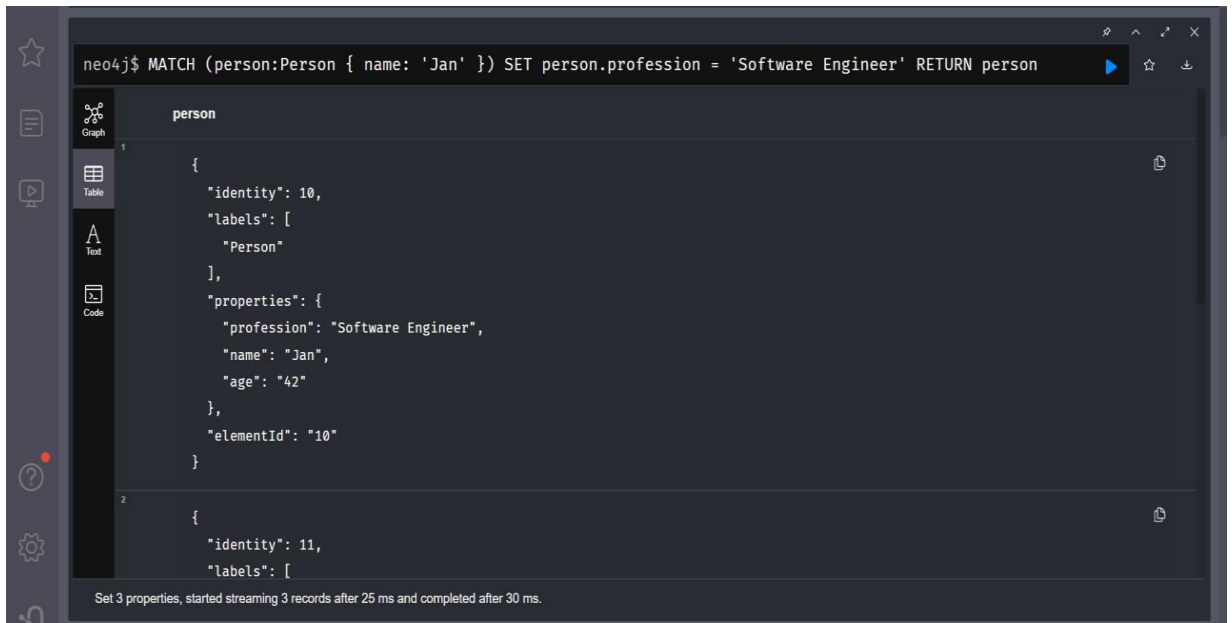
Overview

Node labels

Person (3)

Displaying 3 nodes, 0 relationships.

Jika sudah berhasil maka akan ditampilkan 3 set properties seperti ini



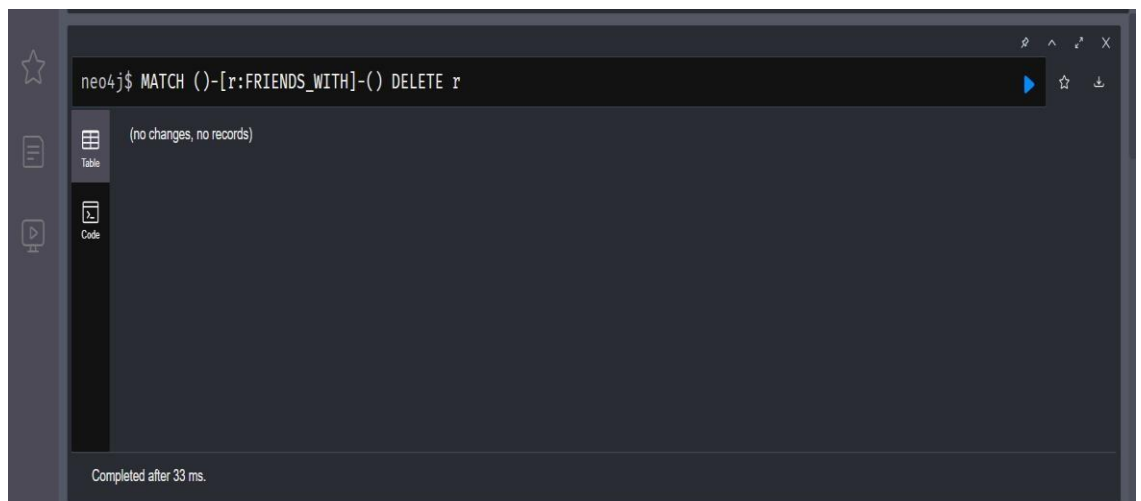
9. Kita dapat **menghapus relationship dan node** dengan cara dibawah ini

➤ **Menghapus relationship**

Gunakanlah syntax untuk menghapus relationship

```
MATCH ()-[r:FRIENDS_WITH]-() DELETE r
```

Maka akan ditampilkan hasilnya seperti ini

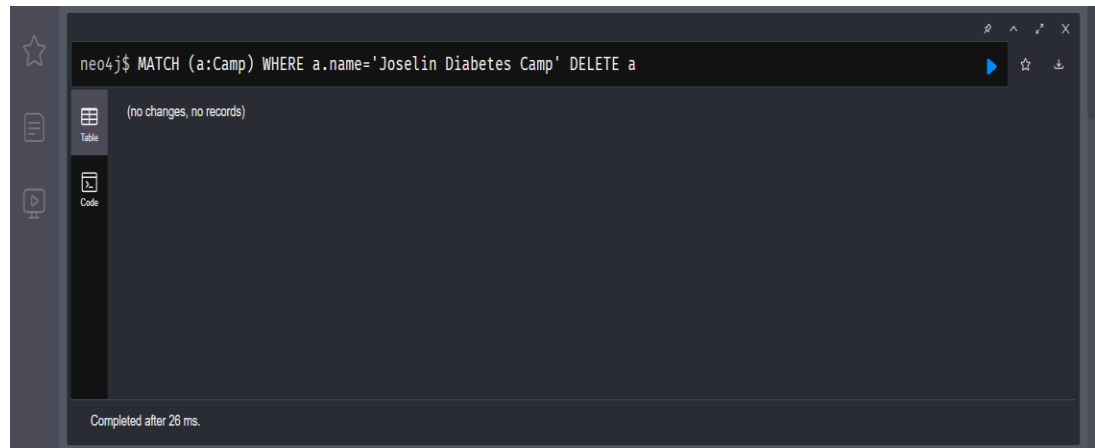


➤ **Menghapus Node**

Gunakanlah syntax untuk menghapus node

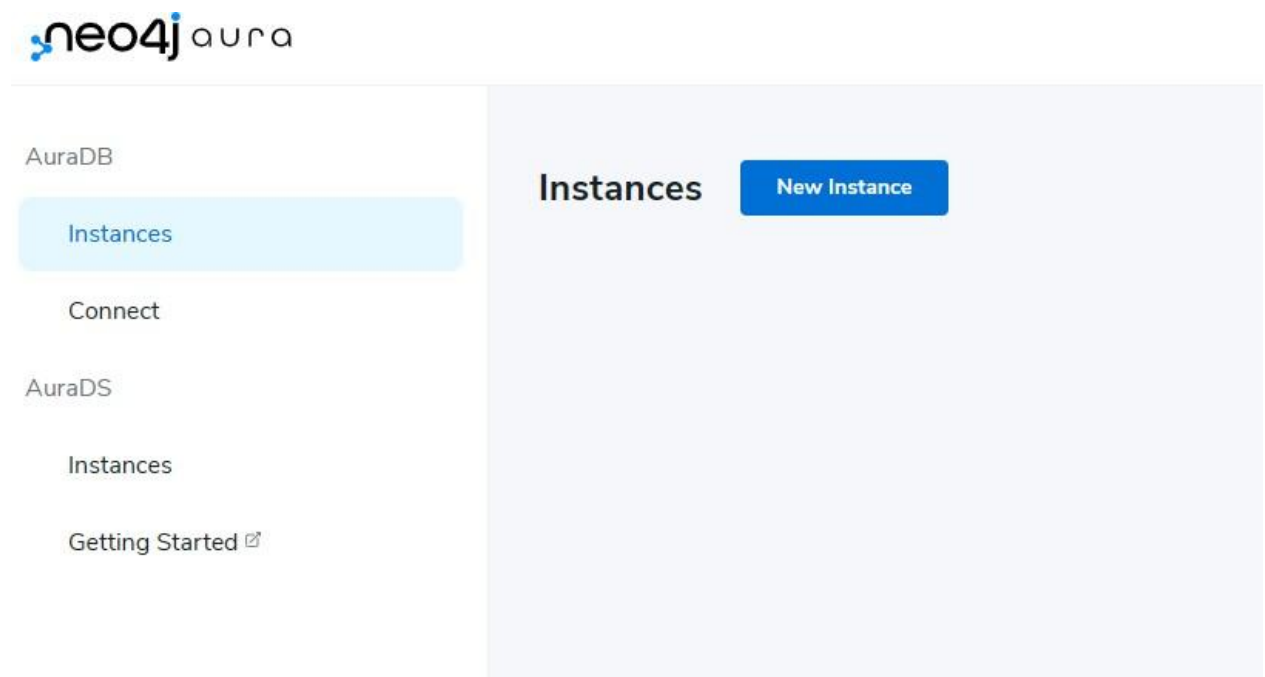
```
MATCH (a:Camp) WHERE a.name='Joselin Diabetes Camp' DELETE a
```

Maka akan ditampilkan hasilnya seperti ini



III. Penggunaan neo4j aura

- Link untuk neo4j <https://neo4j.com/> masuk ke neo4j free for develop masuk ke neo4j aura
- Siapkan akun google masing-masing



- Buat New Instances pilih beginner dan NOTE penting untuk pastikan mendownload password sebelum mengkonfirmasi.

Credentials for Instance01

Username: neo4j

Generated password

c-vPRJvJzCdc-PB343LJT33zs7qOwTPGqRl



Download

We strongly advise changing this initial password.

- ☐ I confirm I have have copied or downloaded the above credentials, as this password will not be available after this point

Continue

- setelah dikonfirmasi klik open



AuraDB

Instances

Connect

AuraDS

Instances

Getting Started

Instances

New Instance

Instance01 Free

Open

Running

Neo4j version 5

Nodes 0 / 200000 (0%)

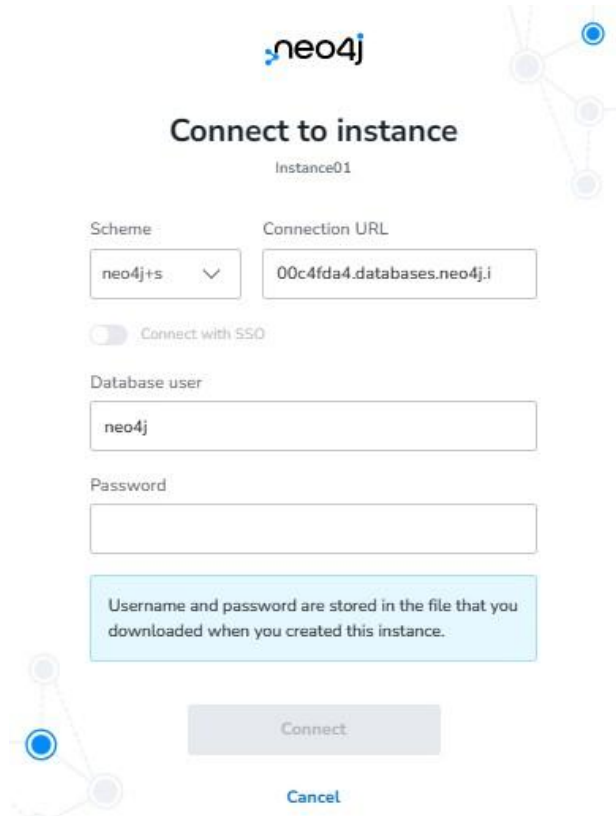
Relationships 0 / 400000 (0%)

Region Singapore (asia-southeast1), GCP

Connection URI neo4j+s://00c4fda4.databases.neo4j.io

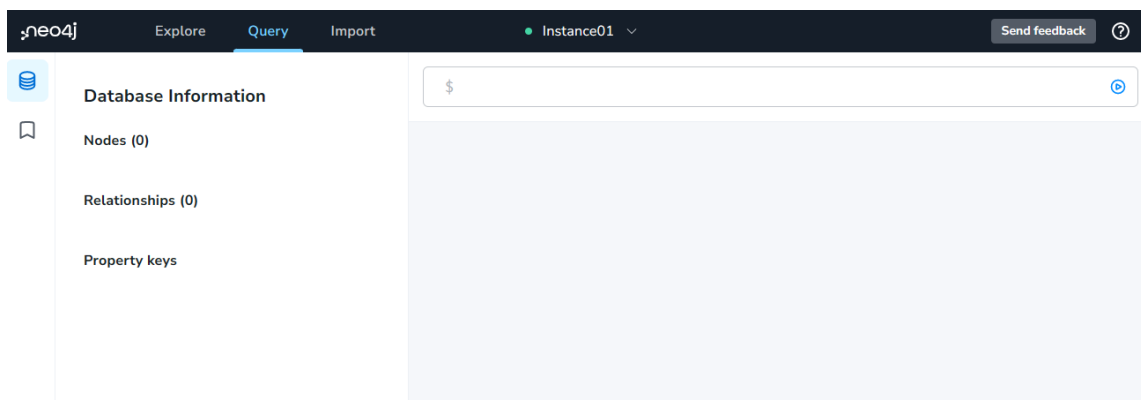


- Siapkan password yang telah didownload untuk tahap dibawah.



The image shows the 'Connect to instance' dialog in the Neo4j desktop application. At the top, the Neo4j logo is displayed. Below it, the title 'Connect to instance' is centered, with 'Instance01' underneath. The dialog contains several input fields: 'Scheme' is set to 'neo4j+s', 'Connection URL' is '00c4fda4.databases.neo4j.i', 'Database user' is 'neo4j', and 'Password' is empty. A toggle switch for 'Connect with SSO' is turned off. A blue information box states: 'Username and password are stored in the file that you downloaded when you created this instance.' At the bottom, there are 'Connect' and 'Cancel' buttons. Decorative graph icons are visible in the corners.

- Copy NEO4J_PASSWORD dari dokumen yang didownload



A. CREATING NODES

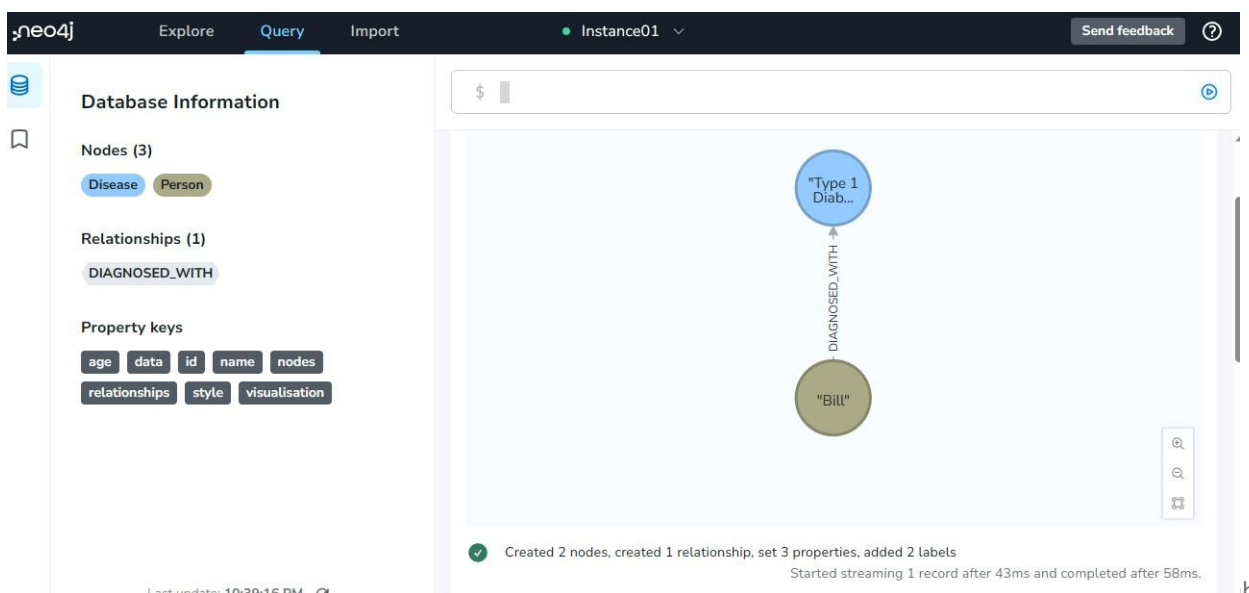
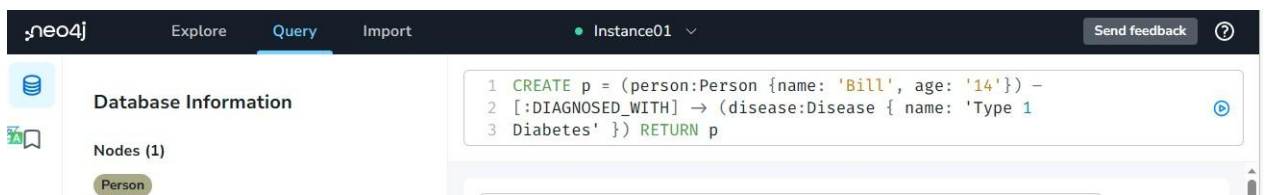


Dengan output:



- Lakukan cara diatas buatlah:
 - node person dengan Samantha dengan umur 32 {masih dalam node person}
 - node untuk camp dengan Joselin Diabetes Camp {node yang berbeda}

B. Manually create nodes with relationships



C. ADDING RELATIONSHIPS

Add a relationship between people nodes

```
1 MATCH (p:Person {name:'Jan'}), (f:Person {name:'Samantha'})
2 CREATE (p)-[:FRIENDS_WITH {since: 2009}]->(f)
```



D. UPDATING NODE PROPERTIES

```
1 MATCH (person:Person { name: 'Jan' })
2 SET person.profession = 'Software Engineer'
3 RETURN person
```

- Setelah di run maka dalam tampilan table akan berubah jan adalah software engineer

Graph **Table** RAW

n

```
{
  identity: 0,
  labels: ["Person"],
  properties: {
    profession: "Software Engineer",
    name: "Jan",
    age: "42"
  },
  elementId: "4:a584c95a-8ca7-4535-98de-4355888bf865:0"
}
```

E. DELETING RELATIONSHIPS AND NODES

- Menghapus relationships

```
$ MATCH ()-[r:FRIENDS_WITH]-() DELETE r
```

base

```
$ MATCH ()-[r:FRIENDS_WITH]-() DELETE r
```



Deleted 1 relationship

Completed after 42ms

- Menghapus node

```
1 MATCH (a:Camp)
2 WHERE a.name='Joselin Diabetes Camp'
3 DELETE a
```

