

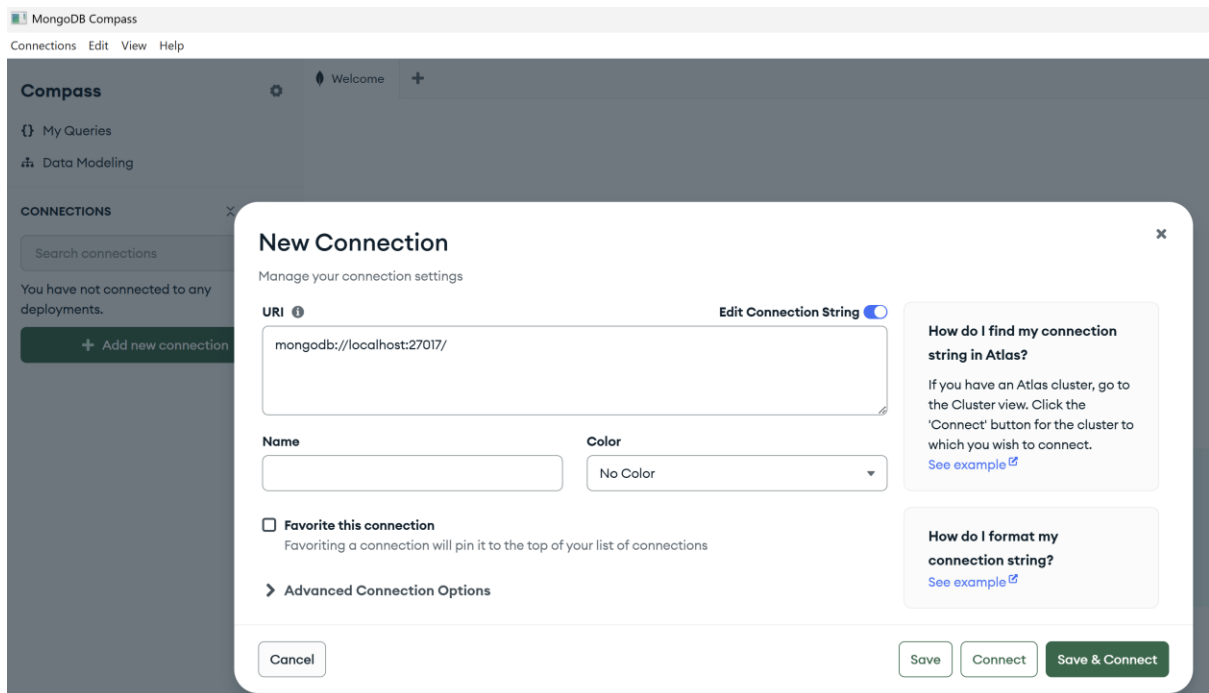
Lab 3

Part 1

First we have to start the mongo database by running the docker command

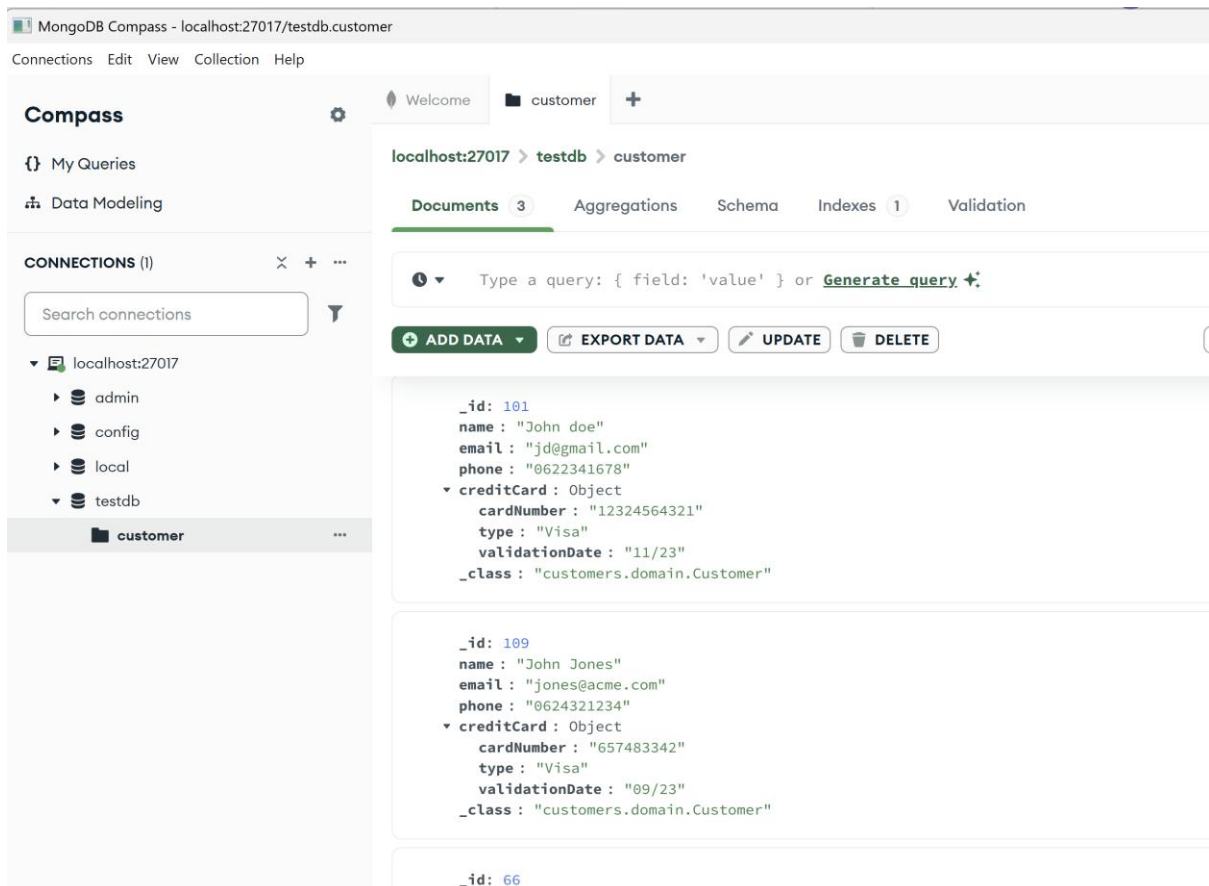
```
docker run --name mongodb -d -p 27017:27017 mongo:latest
```

Then download **MongoCompass** from <https://www.mongodb.com/try/download/compass>
Install and start MongoCompass



Click the **Save&Connect** button.

Run the given **Lesson3SpringMongoDemo** application and see that the application saves some customers with creditcards in the **testdb** database.



Modify the given **Lesson4SpringMongoDemo** application so that the application stores students in the database

A Student contains the attributes name, phoneNumber and email.

A student has also an address.

Create a class Address with the attributes street, city and zip.

In the application add 5 students in the database.

Then perform the following queries:

- Get all students
- Get all students with a certain name
- Get a student with a certain phoneNumber
- Get all students from a certain city

Check the data in the collection using Mongo compass.

Part 2

Go to <https://www.datastax.com/pricing/astra-db>

DATASTAX an IBM Company PRODUCTS PRICING SOLUTIONS STORIES RESOURCES Docs Contact Us Sign In **TRY FOR FREE**

THIS SITE WILL MIGRATE OVER TO IBM.COM AS DATASTAX HAS BEEN ACQUIRED BY IBM

Free	Pay As You Go	Enterprise
No Credit Card Required	Only Pay for What You Use	For Maximum Flexibility and Scale
80GB of storage	EVERYTHING IN FREE, PLUS:	EVERYTHING IN PAY AS YOU GO, PLUS:
20 million read/write	Scale automatically as your app grows	Volume discounts with annual commits
Build and deploy on AWS	Build and deploy on AWS, Azure or Google Cloud	Health checks for optimal sizing and performance
Data API for seamless programmatic access to your data	Multi-region databases	Provisioned capacity consumption model
Hybrid-Vector Search for maximum accuracy for agents/AI apps	Secure private networking	Access to AI partner credits
	Chat and community support	24x7 enterprise support
	Option to upgrade to 24x7 Enterprise Support (requires prepaid commitment)	Option to add Premium Support for 15-minute response times and other premium features
START FOR FREE	GET STARTED	GET AN ESTIMATE

Select the **Free** option

DATASTAX

Start Today with \$300/Year Credit

Sign Up with GitHub

Sign Up with Google

OR

First Name
Rene

Last Name
de Jong

Company Email
rene@ictintelligence.nl

☒ I agree to DataStax's [Privacy Notice](#), [MSA](#) and [Terms of Service](#)

Get Started Today

Enter your data and click the **Get Started Today**

You need to verify your account in your email, and then you need to enter a password.

Welcome to Astra

Knowing you better will help us improve your experience.

How do you plan to use Astra?

Work

Personal

Both

How would you describe your role?

App Developer

AI Engineer

ML Engineer

Data Engineer

Cloud Engineer

Architect

Manager/Director

CTO/VP

Student or Learner

Other

What is your primary programming language?

Python

JavaScript/TypeScript

Java

Go

PHP

Other

I don't code

How did you hear about Astra?

Google/Search

Blog/Tech Article

LinkedIn

Referred by someone

Twitter/X

YouTube

Partner/Integration

Workshop

Event/Conference

Other

Continue to Astra →

Click the **Continue to Astra** button.

Preview

Generate embeddings from unstructured data directly in Astra ✨

DATAStax

rene@ictintelligence.nl

Home

Databases

Streaming

Billing

Tokens

Integrations

Settings

Sample Apps

Welcome to Astra, Rene 🌟

Generate embeddings direct from Astra DB

Automate embedding generation directly from unstructured data in your Astra DB collection with OpenAI, Azure OpenAI, and NVIDIA integrations. Then, test and tune for relevancy with vector text search.

Add Integrations →

Read Docs ↗

What's New

Enhance your AI stack with Langflow 🌐

Simplify your GenAI app development with Langflow, now part of DataStax. Learn how integrating Langflow with Astra DB can enhance your AI stack's capabilities.

Try Langflow ↗

Data API + MongoDB API Compatibility 🌐

Transition your MongoDB experience to Astra DB smoothly with our updated Data API. Build robust, scalable applications faster and with less overhead.

View API Updates ↗

Effortless Agents with Astra + Vertex AI 🌐

The new integration between DataStax Astra DB and the new Agent Builder for Vertex AI Search empowers you to leverage your Astra DB data to build next-generation search and conversational AI applications.

View Announcement ↗

Quick Access

Create a Database →

Generate Token →

Invite Your Team →

Click **Create a Database**

Create Database

ESC

Select a deployment type

Serverless (Vector) **Recommended**
An all-in-one database solution, optimized for Vector and Generative AI workloads

Serverless (Non-Vector)
A more traditional database solution without any of our new vector capabilities

Configuration

Database name*

demo database

Give it a memorable name – this can't be changed later.

Provider*

aws Amazon Web Services

Region*

us-east-2

Upgrade to get more regions and premium features

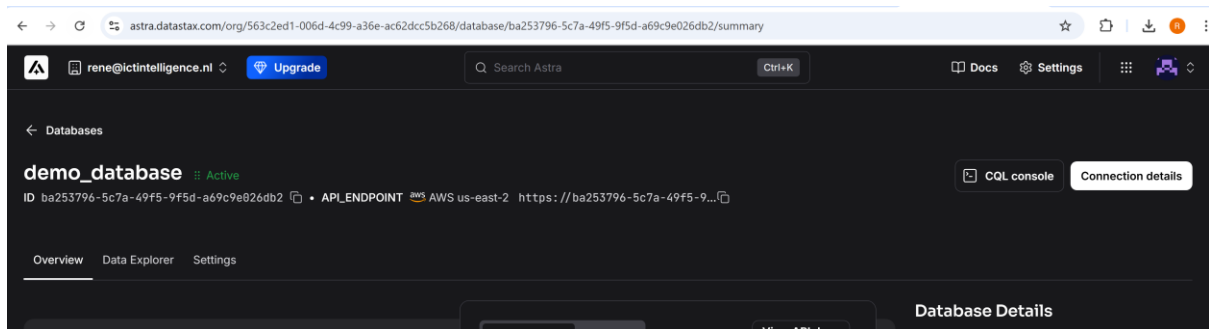
Upgrade

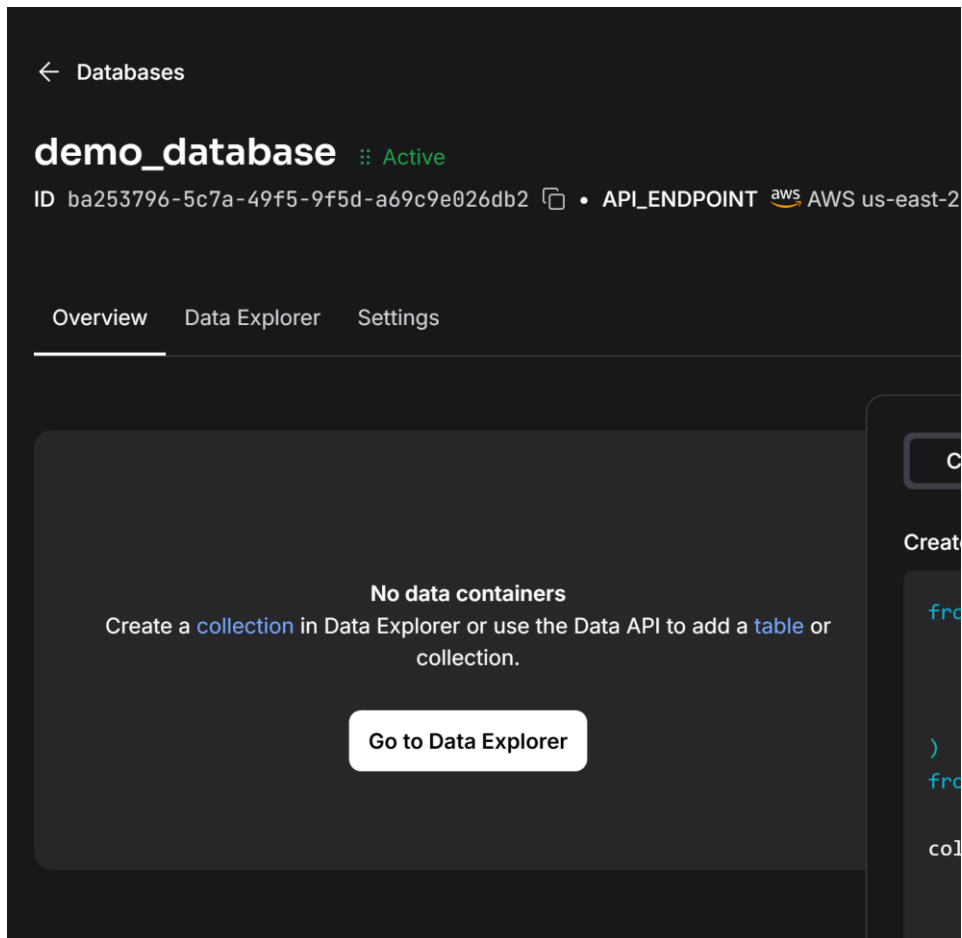
Cancel

Create Database

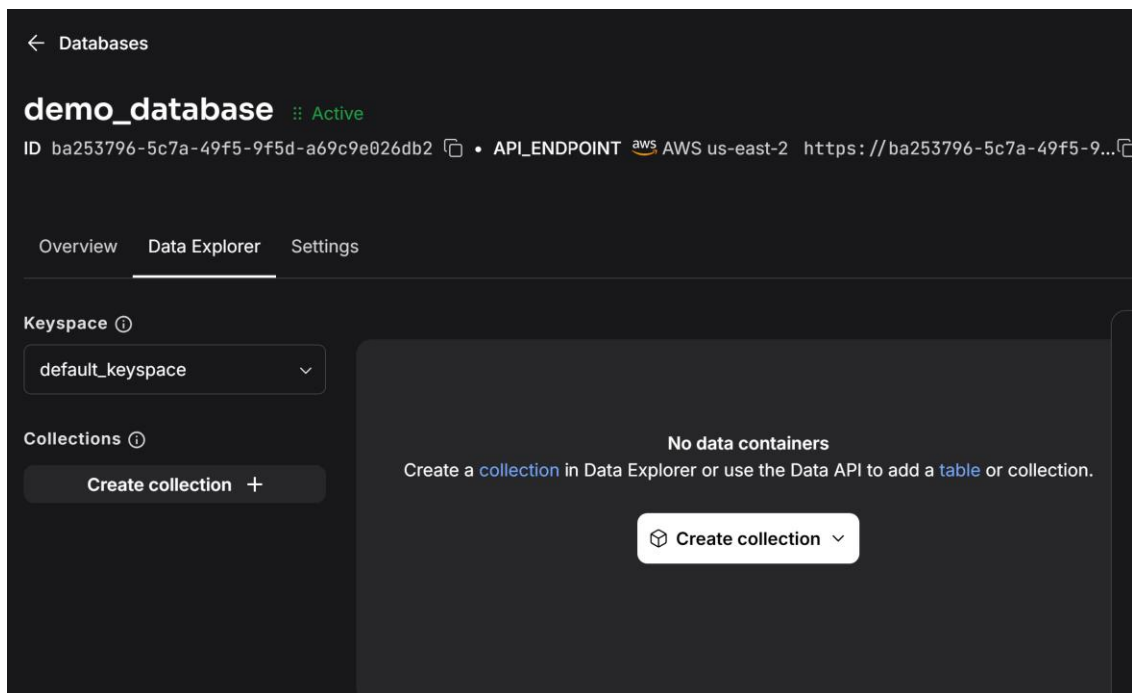
Enter the database name and select a region. Then click **Create Database**

Wait till the database is created





Click the **Go to Data Explorer** button.



Select the namespace drop down list and select **create a new namespace**

Create Namespace

ESC ✕

Namespaces serve as containers for organizing and managing data within a database.

Namespace Name*

Alphanumeric characters only, no spaces or special characters

Cancel

Add Namespace

Create the **demo** namespace

The screenshot shows the 'demo_database' namespace page. At the top, it says 'demo_database' with a status 'Active'. Below this, the ID 'ba253796-5c7a-49f5-9f5d-a69c9e026db2' is shown along with the API endpoint 'AWS us-east-2' and a URL. There are three tabs: 'Overview', 'Data Explorer', and 'Settings'. Under 'Overview', there is a 'Keyspace' dropdown menu set to 'demo'. Below that is a 'Collections' section with a 'Create collection +' button. The main area of the page says 'No data containers' and 'Create a collection in Data Explorer or use the Data API to add a table'. There is a 'Create collection' button with a dropdown arrow.

The screenshot shows the Astra Data Explorer interface. The browser address bar shows the URL 'astra.datastax.com/org/563c2ed1-006d-4c99-a36e-ac62dcc5b268/database/ba253796-5c7a-49f5-9f5d-a69c9e026db2/summary'. The page has a header with the Astra logo, a user profile 'rene@ictintelligence.nl', an 'Upgrade' button, a search bar, and links for 'Docs', 'Settings', and a menu. The main content area shows the 'demo_database' namespace with its ID and API endpoint. There are buttons for 'CQL console' and 'Connection details'. At the bottom, there are tabs for 'Collections', 'Tables', and 'View API docs', and a 'Database Details' section.

Now click the **CQL Console** button

Preview

Generate embeddings from unstructured data directly in Astra ⚡

DS

🔧

🏠

📄

🔍

🔑

🔗

⚙️

🔧

🔧

🔧

⋮

🔧

Connect to your CQL Console

Use the [CQL console](#) to interact with your database through Cassandra Query Language (CQL).

Full documentation for CQL commands:
https://docs.datastax.com/en/dse/6.8/cql/cql_reference/cql_commands/cqlCommandsTOC.html

```

token@cqlsh> DESCRIBE keyspaces;

system_auth      default_keyspace  demo      system_virtual_schema
system_schema    system            datastax_sla
data_endpoint_auth system_traces     system_views

token@cqlsh> use demo;
token@cqlsh:demo> select * from employees;

key | array_contains | array_size | doc_json | exist_keys | query_bool_values | query_dbl_values | query_null_values | query_text_values | query_timestamp_values | tx_id
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
(0 rows)

token@cqlsh:demo> CREATE TABLE emp(
...   emp_id int PRIMARY KEY,
...   emp_name text,
...   emp_city text,
...   emp_sal varint,
...   emp_phone varint
... );
token@cqlsh:demo> select * from emp;

emp_id | emp_city | emp_name | emp_phone | emp_sal
-----+-----+-----+-----+-----
1 | Amsterdam | John | 9848022338 | 50000
2 | Berlin | Mary | 9848022339 | 40000
3 | London | Frank | 9848022330 | 45000

(3 rows)

token@cqlsh:demo> INSERT INTO emp (emp_id, emp_name, emp_city,
...   emp_phone, emp_sal) VALUES(1,'John', 'Amsterdam', 9848022338, 50000);
token@cqlsh:demo> INSERT INTO emp (emp_id, emp_name, emp_city,
...   emp_phone, emp_sal) VALUES(2,'Mary', 'Berlin', 9848022339, 40000);
token@cqlsh:demo> INSERT INTO emp (emp_id, emp_name, emp_city,
...   emp_phone, emp_sal) VALUES(3,'Frank', 'London', 9848022330, 45000);
token@cqlsh:demo> select * from emp;

emp_id | emp_city | emp_name | emp_phone | emp_sal
-----+-----+-----+-----+-----
1 | Amsterdam | John | 9848022338 | 50000
2 | Berlin | Mary | 9848022339 | 40000
3 | London | Frank | 9848022330 | 45000

(3 rows)

token@cqlsh:demo> SELECT emp_name, emp_sal from emp;

emp_name | emp_sal
-----+-----
John | 50000
Mary | 40000
Frank | 45000

(3 rows)

```

Now write the following commands in the console:

DESCRIBE keyspaces;

USE demo;

CREATE TABLE ratings_by_user (email TEXT, title TEXT, year INT, rating INT, PRIMARY KEY ((email), title, year));

Now insert the following data in the table:

```
INSERT INTO ratings_by_user (email, title, year, rating)
VALUES ('joe@datastax.com', 'Alice in Wonderland', 2010, 9);
INSERT INTO ratings_by_user (email, title, year, rating)
VALUES ('joe@datastax.com', 'Edward Scissorhands', 1990, 10);
INSERT INTO ratings_by_user (email, title, year, rating)
VALUES ('jen@datastax.com', 'Alice in Wonderland', 2010, 10);
INSERT INTO ratings_by_user (email, title, year, rating)
VALUES ('jen@datastax.com', 'Alice in Wonderland', 1951, 8);
```

Perform the following query:

```
SELECT * FROM ratings_by_user;
```

Write the following queries:

Retrieve all data from the user with email 'joe@datastax.com'

```
SELECT * FROM ratings_by_user
WHERE email = 'joe@datastax.com';
```

Retrieve all rating data from the user with email 'joe@datastax.com' for the movie 'Alice in Wonderland' from the year 2010

```
SELECT * FROM ratings_by_user
WHERE email = 'joe@datastax.com'
  AND title = 'Alice in Wonderland'
  AND year = 2010;
```

Now try to get the data for all movies with a rating of 10:

```
SELECT * FROM ratings_by_user
WHERE rating= 10;
```

Notice that this query is not possible because rating is not part of the PK.

Now create the following table:

```
CREATE TABLE ratings_by_movie (
  title TEXT,
  year INT,
  email TEXT,
  rating INT,
  PRIMARY KEY ((title, year), email)
);
```

And populate the table with the following data:

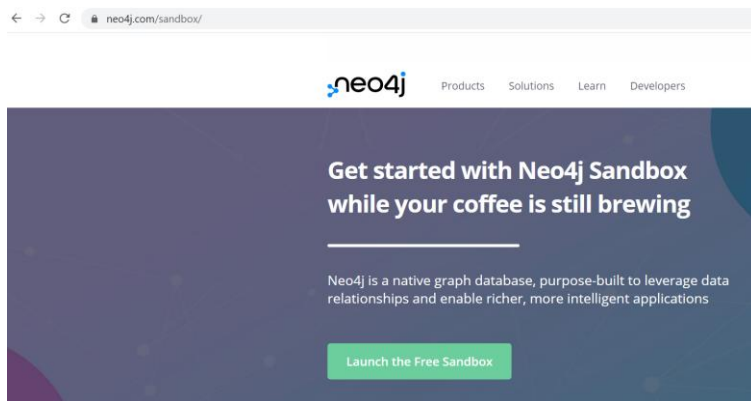
```
INSERT INTO ratings_by_movie (title, year, email, rating)
VALUES ('Alice in Wonderland', 2010, 'jen@datastax.com', 10);
INSERT INTO ratings_by_movie (title, year, email, rating)
VALUES ('Alice in Wonderland', 2010, 'joe@datastax.com', 9);
INSERT INTO ratings_by_movie (title, year, email, rating)
VALUES ('Alice in Wonderland', 1951, 'jen@datastax.com', 8);
INSERT INTO ratings_by_movie (title, year, email, rating)
VALUES ('Edward Scissorhands', 1990, 'joe@datastax.com', 10);
```

Write the query to select all ratings from 'Alice in Wonderland' from 2010

```
SELECT * FROM ratings_by_movie
WHERE title = 'Alice in Wonderland'
AND year = 2010;
```

Part 3

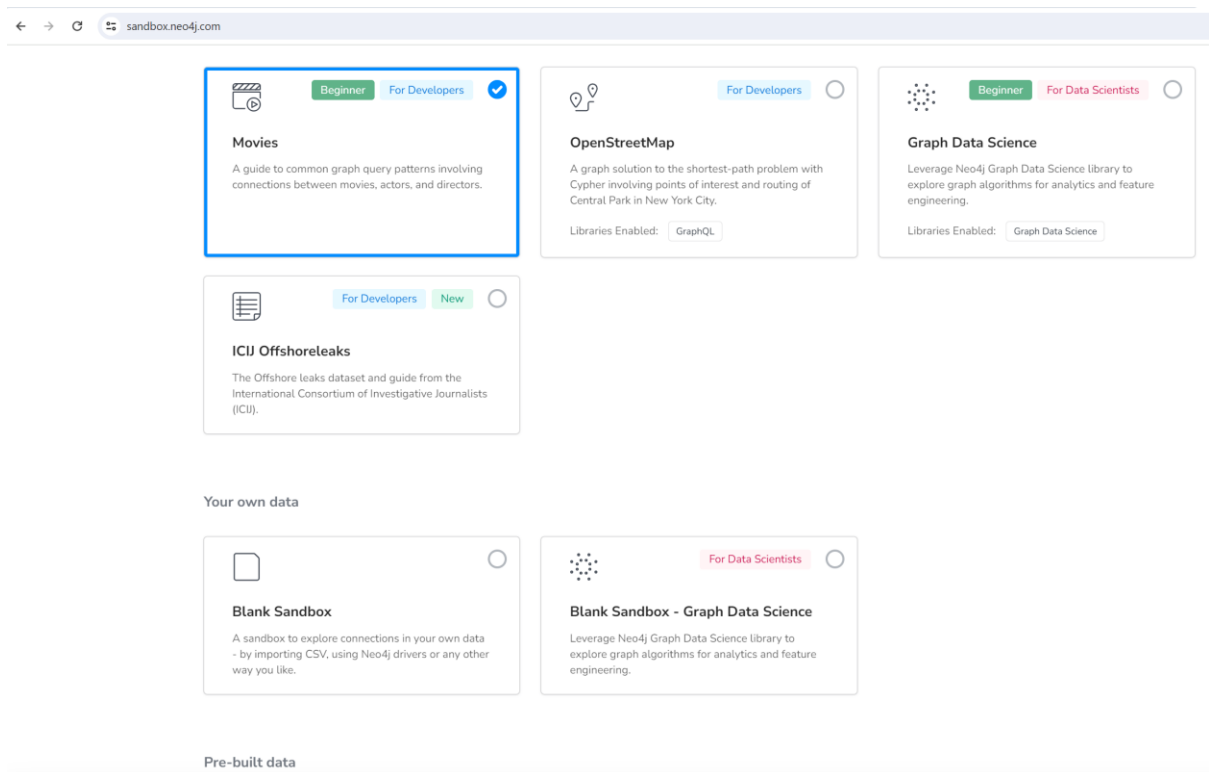
Go to <https://neo4j.com/sandbox/>



Click the **Launch the free Sandbox** button.

Sign up or login with Google, LinkedIn or Twitter

Setup your account and login.



Choose **Blank Sandbox** and click **Create**



+ New Project

Name	Status	
Blank Sandbox	Running Expires in about 3 days	Open

Now **Open** your project.

\$:server connect

Connect to Neo4j

Database access might require an authenticated connection

Connect URL

neo4j+s:// ee029abcb7955ae20f0630e7149c35bf.bolt.neo4jsandbox.com:443

Authentication type

Single Sign On

[Sandbox Login](#)

Click **Sandbox Login**

Login with your username and password.

neo4j\$

To help make Neo4j Browser better we collect information on product usage. Review your settings at any time.

\$:play start

Getting started with Neo4j Browser

Neo4j Browser user interface guide

[Get started](#)

Try Neo4j with live data

A complete example graph that demonstrates common query patterns.

Actors & movies in cross-referenced pop culture.

[Open guide](#)

Cypher basics

Intro to Graphs with Cypher

What is a graph database?
How can I query a graph?

[Start querying](#)

Copyright © Neo4j, Inc 2002–2024

\$:server status

Connection status

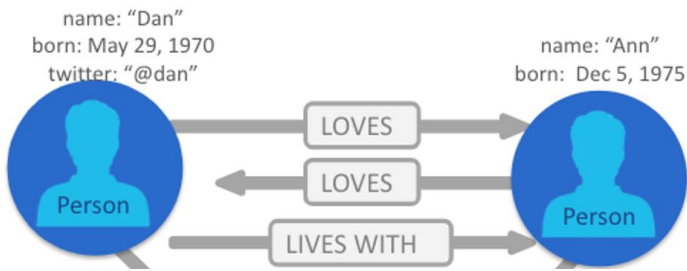
This is your current connection information.

You are connected as user **rene@ictintelligence.nl**

to **neo4j+s://ee029abcb7955ae20f0630e7149c35bf.bolt.neo4jsandbox.com:443**

Connection credentials are stored in your web browser.

We are going to implement the following graph:



Enter the following Cypher statement in the sandbox:

CREATE

```
(:Person {name:'Dan', born: 'May 29, 1970', twitter: '@dan'}),  
(:Person {name:'Ann', born: 'Dec 5, 1975'})
```

Run the statement on the Neo4j command line:

```
1 CREATE
2 (:Person {name:'Dan', born: 'May 29, 1970', twitter: '@dan'}),
3 (:Person {name:'Ann', born: 'Dec 5, 1975'})
4
```

Added 2 labels, created 2 nodes, set 5 properties, completed after 29 ms.

Table

Code

Now query all nodes with: **MATCH (n) RETURN n**

neo4j\$ MATCH(n) RETURN n

Overview

Node labels

Person (2)

Displaying 2 nodes, 0 relationships.

Now we are going to create the following relationships:

```
MATCH (pdan: Person), (pann: Person)  
WHERE pdan.name = 'Dan' AND pann.name = 'Ann'  
CREATE (pdan)-[:LOVES]->(pann)
```

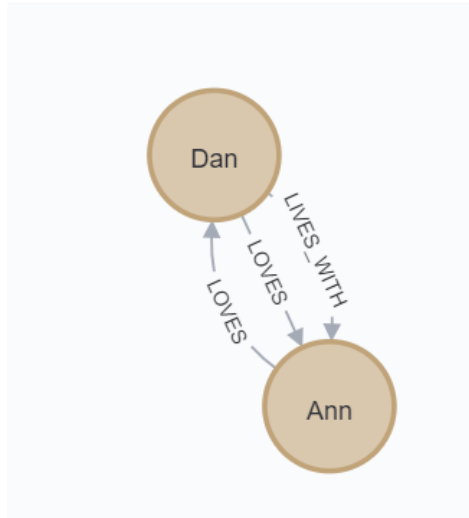
```
MATCH (pdan: Person), (pann: Person)  
WHERE pdan.name = 'Dan' AND pann.name = 'Ann'  
CREATE (pann)-[:LOVES]->(pdan)
```

```

MATCH (pdan: Person), (penn: Person)
WHERE pdan.name = 'Dan' AND penn.name = 'Ann'
CREATE (pdan)-[:LIVES_WITH]->(penn)

```

This should give the following graph:



Now we want to know the person where Dan lives with:

```

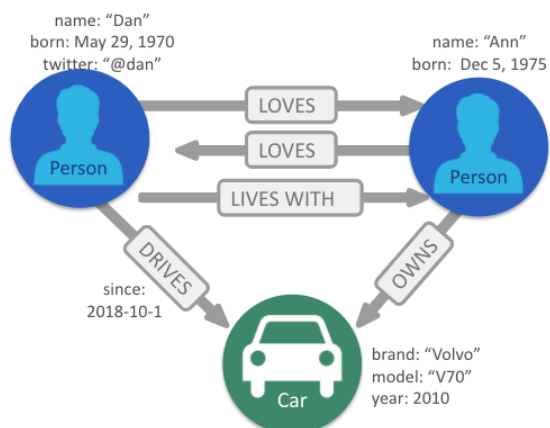
MATCH (p:Person {name : 'Dan'}) -[rel:LIVES_WITH]->(partner:Person)
RETURN partner.name

```

```
neo4j$ MATCH (p:Person {name : 'Dan'}) -[rel:LIVES_WITH]->(partner:Person) RETURN partner.name
```

partner.name
"Ann"

Now extend the graph as follows:



Write the following queries:

1. Find all persons who own a Volvo car
2. Find the Car that is driven by Dan
3. Find all persons who love someone that owns a car.

Now create the following graph:

Nodes:

- 3 people: Alice, Bob, and Carol
- 2 companies: NeoTech and DataWorks

Relationships:

- Alice KNOWS Bob
- Bob KNOWS Carol
- Alice WORKS_AT NeoTech
- Bob WORKS_AT DataWorks
- Carol WORKS_AT NeoTech

Then create the following queries, and check if they work correctly:

1. Find who works at which company
2. Find all people who work at NeoTech
3. Find all people who Alice knows

Part 4

Suppose you need to store the following order in the database:

Ordernumber:122435

Orderdate 11/09/2021

Customer name: Frank Brown

Customer email: fbrown@gmail.com

Customer phone: 0623156543

Total price : 5160.00

quantity	Product number	Product name	price
2	A546	IPhone 12	980.00
4	S333	Samsung Galaxy 12S	800.00

1. Draw the tables including data that you need to store this order in a relational database.
2. Draw the collections including data that you need to store this order in a mongo database.
3. Draw the tables including data that you need to store this order in a cassandra database if you are interested in orders by customer.
4. Draw the database structure including data that you need to store this order in a neo4j database.

What to hand in?

1. A zip file of part 1
2. A screenshot of part 2
3. Multiple screenshots of part 3
4. A PDF of part 4