DATABASE

PROJECT

BY
NAME: GAD NWACHUKWU OLUNGWEONWI

KALTURAL LINK: https://video.northampton.ac.uk/media/Kaltura+Capture+recording+-+May+21st+2023%2C+2A46A33+pm/1_051kcflc

Table of Contents

ASSIGNMENT REPORT 1

1 ANALYSIS OF THE DATASET AND THE DESIGN PROCESS 3
1.1 DATABASE NAME – **SCHOOL** 3
2 BOTH MODELS WITH GOOD EXPLANATIONS/JUSTIFICATION 4
2.1 PURPOSE OF DATABASE 4
3 THE PROCESS OF DATABASE CREATION 5
3.1 THE TOOL USED TO CREATE THE DATABASE IS MYSQL WORKBENCH. 5
3.2 IMPORT THE CSV DATASETS 5
3.3 THE "AWARDTITLE" TABLE 7
3.4 THE "MODULES" TABLE 7
3.5 THE "COURSES" TABLE: 8
3.6 THE PICTURE BELOW SHOWS THE ERD DIAGRAM FOR THE RELATIONAL MODEL; 8
4 CAREFULLY DESIGNED SQL AND CYPHER COMMANDS TO ANSWER THE FOLLOWING QUERIES 9
4.1 DISPLAY THOSE MODULES RUNNING IN BOTH SEMESTERS 9
4.2 DISPLAY ALL MODULES WITH AT LEAST A DESIGNATED MODULE. QUERY 10
4.3 WHICH COURSE HAS THE MOST PREREQUISITES 11
4.4 DISPLAY THOSE MODULES THAT APPEAR ON A SINGLE COURSE: QUERY: 12
4.5 DOES THE COMPUTER SCIENCE COURSE HAS UNBLANCED WORK LOAD IN BOTH SEMESTERS 13
5 ASSIGNMENT QUERIES USING GRAPH DATABASE MODEL (NEO4J) 14
5.1 N EO4J :SCHOOL 15
5.2 THE CSV FILES WERE LOADED ON THE SCHOOL DATABASE IN NO RESPECTIVE ORDER AS SHOWN BELOW IN THE NEO4J BROWSER: 17
5.3 THE FOLLOWING RELATIONSHIPS WERE CREATED AS FOLLOWS: 20
6 NEO4J QUERY 22
6 .1 DISPLAY THOSE MODULES RUNNING IN BOTH SEMESTERS: 22
6.2 DISPLAY ALL MODULES WITH ALEAST A DESIGNATED MODULE: 23
6.3 WHICH COURSE HAS MOST HAS MOST PRE_REQUISITE 24
6.4 WHICH COURSE APPEAR ON A SINGLE COURSE: 25
6.5 DOES THE COMPUTER SCIENCE COURSE HAS UNBLANCED WORK LOAD IN BOTH SEMESTERS 26
6.6 COMPARISON/REFLECTION OF RELATIONAL AND GRAPHICAL APPROACHES: 28

1 Analysis of the dataset and the design process

1.1 Database name – **school**

Tables

- ai
- ws
- cse
- cs
- bc
- cse
- modules
- courses

Columns and Data Types

- **ai, ws, cse, cs, bc, cn, se**
  - ➢**course_code** – Integer(foreign key and related to column course_code on table courses)
  - ➢**mod_code** – Varchar(Foreign key and related to column mod_code on table modules)
  - ➢**requirement** – Varchar

- **modules**
  - ➢**mod_code** – Varchar(Primary Key)
  - ➢**old_code** – Varchar
  - ➢**title** – Varchar
  - ➢**credits** – Integer
  - ➢**level** – Integer
  - ➢**semester** – Varchar
  - ➢**pre_requisite** – Varchar
  - ➢**module_leader** – Varchar

- **courses**
  - ➢**course_code** – Integer(Primary Key: AutoIncrement)
  - ➢**award_title** – Varchar
  - ➢**level** - Varchar

2 Both models with good explanations/justification

2.1 Purpose of Database

The purpose of the" School" database is to store and generate information about course under computer information and technology.

Courses table showcase the award given to students that study each of the 7 available courses and make us to understand each course is eligible to undergraduate students.

Modules table showcases the module attached to each of the courses and the semesters in which different modules are taken. This table also shows the leader of each module.

AI table shows all module under the Artificial Intelligence course and also their individual requirements.

BC table shows all modules under the Business Computing course and also their individual requirements.

CN table shows all modules under the Computer Networks Engineering course and also their individual requirements.

CS table shows all modules under the Computer Science course and also their individual requirements.

CSE table shows all modules under the Computer Engineering course and also their individual requirements.

SE table shows all modules under the Software Engineering course and also their individual requirements.

WS table shows all modules under the Web Development & Cyber Security course and also their individual requirements.

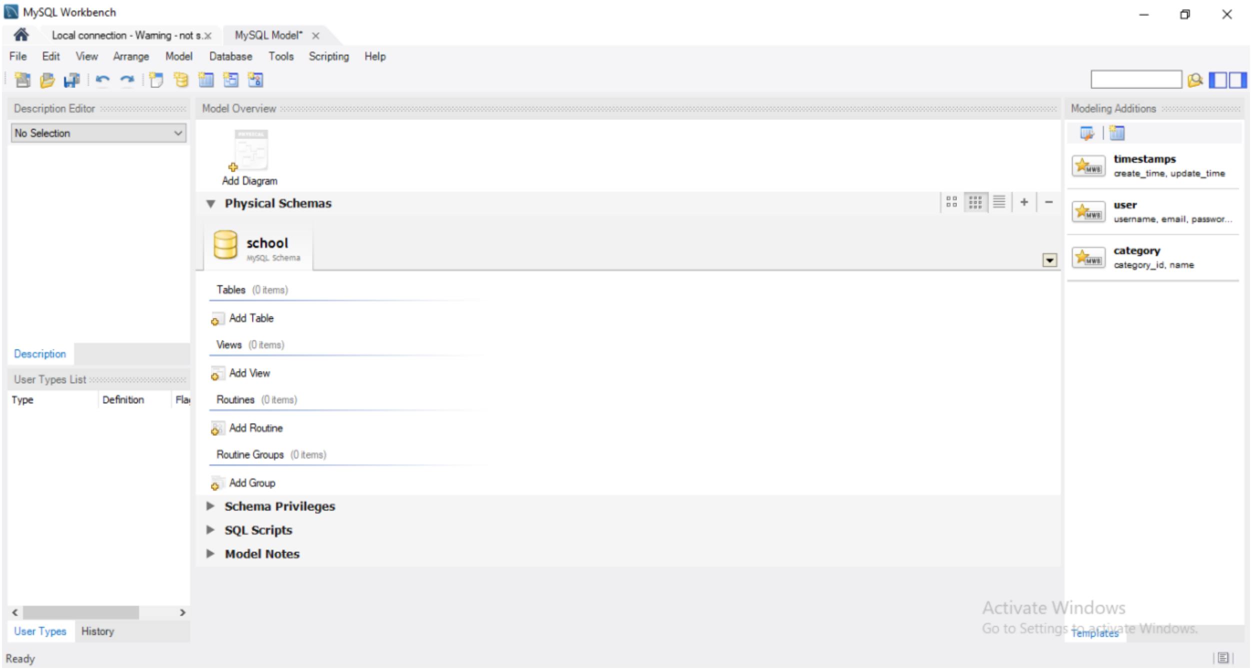
3 The process of database creation

3.1 The tool used to create the database is MySQL workbench.

The following are the steps followed in creating the Database.

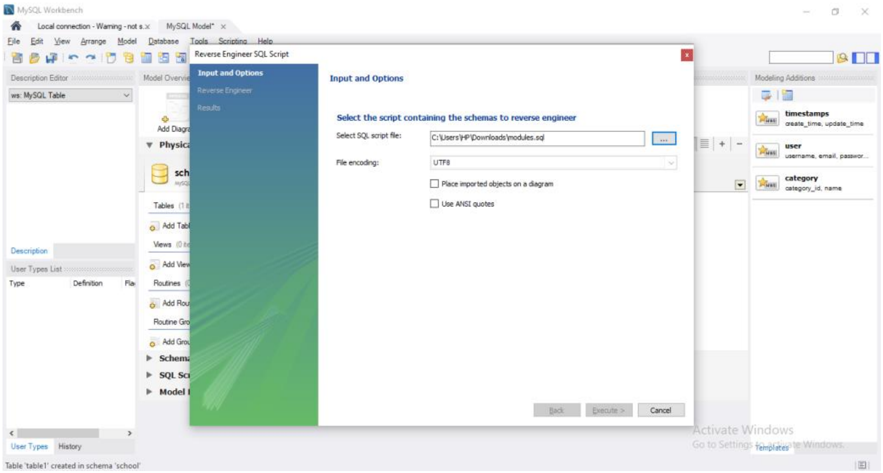
1) **Create a Database called “school”**

   The following screenshot shows the database created using the MySQL workbench client.



3.2 Import the CSV datasets

The following screenshots show the process of importing a csv dataset to create a table using the MySQL workbench client.



Input and Options

Select the script containing the schemas to reverse engineer

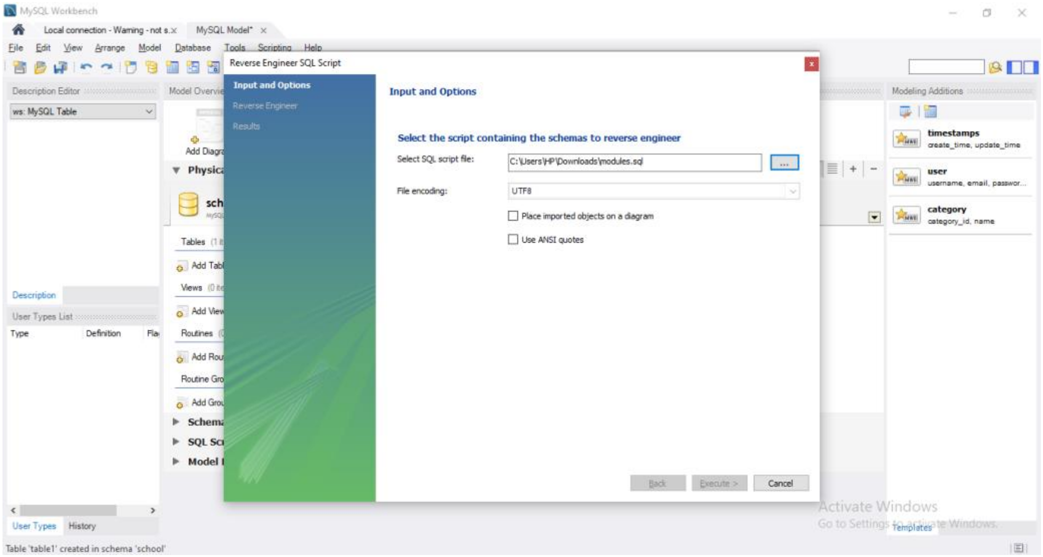
Select SQL script file: C:\Users\HP\Downloads\modules.sql

File encoding: UTF8

□ Place imported objects on a diagram

□ Use ANSI quotes

Table 'table1' created in schema 'school'



Reverse Engineer SQL Script

Input and Options

Select the script containing the schemas to reverse engineer

Select SQL script file: C:\Users\HP\Downloads\modules.sql

File encoding: UTF8

□ Place imported objects on a diagram

□ Use ANSI quotes

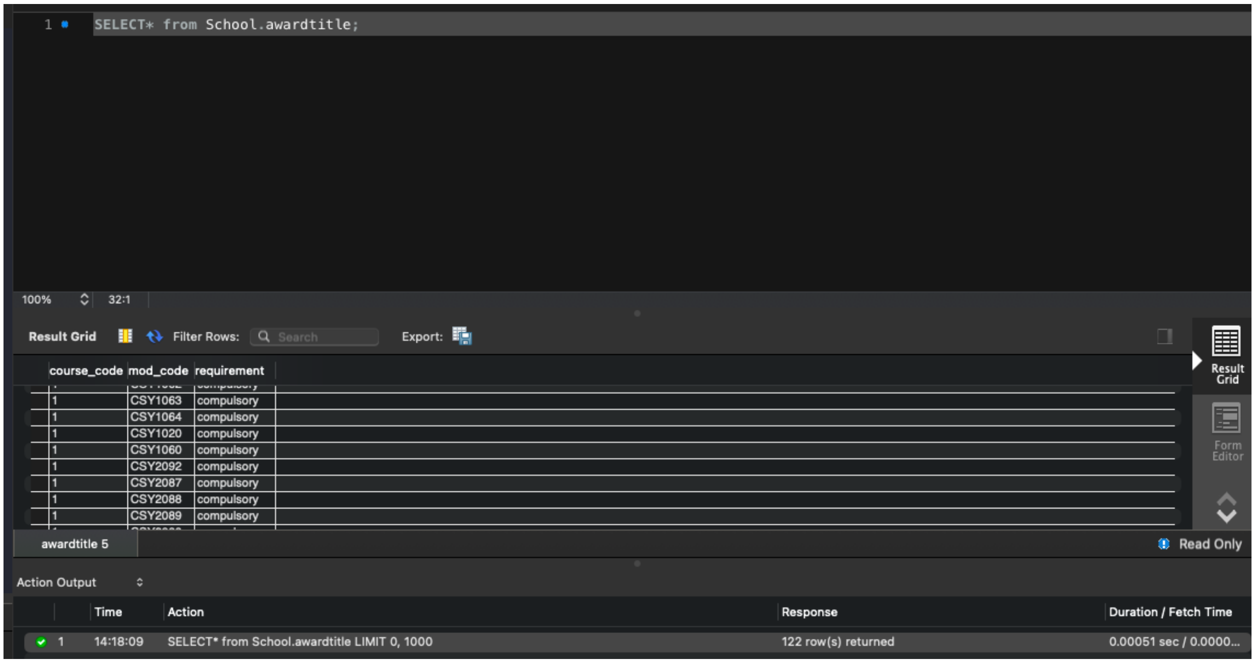
Table 'table1' created in schema 'school'

I believe that normalizing the tables is an important step in database design to improve optimization and reduce complex queries. By breaking down the original nine tables into three separate tables, I have simplified the structure and made it easier to manage the data. Here is a list of the three new tables:

The “awardtitle” Table
The “modules” Table
The “courses” Table:

By breaking the original nine tables into three separate tables, I have reduced the complexity of the database and made it easier to manage the data. This will ultimately result in better optimization and reduced complex queries when working with the data.

3.3 The “awardtitle” Table

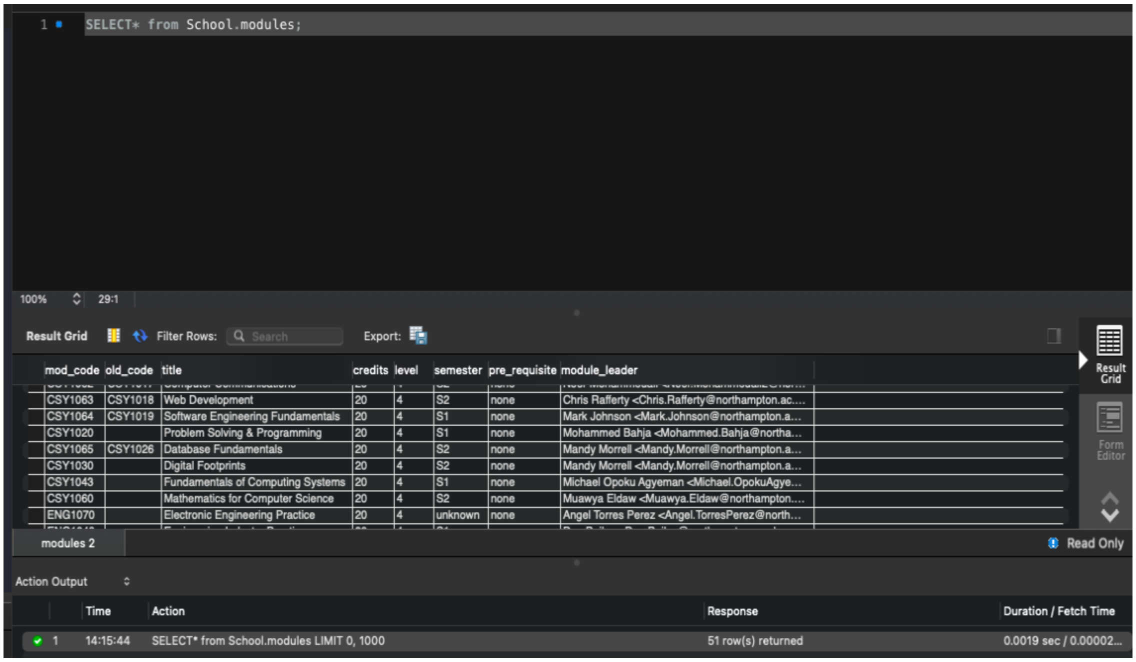


```
SELECT* from School.awardtitle;
```

course_code	mod_code	requirement
1	CSY1063	compulsory
1	CSY1064	compulsory
1	CSY1020	compulsory
1	CSY1060	compulsory
1	CSY2092	compulsory
1	CSY2087	compulsory
1	CSY2088	compulsory
1	CSY2089	compulsory

	Time	Action	Response	Duration / Fetch Time
1	14:18:09	SELECT* from School.awardtitle LIMIT 0, 1000	122 row(s) returned	0.00051 sec / 0.0000...

3.4 The “modules” Table

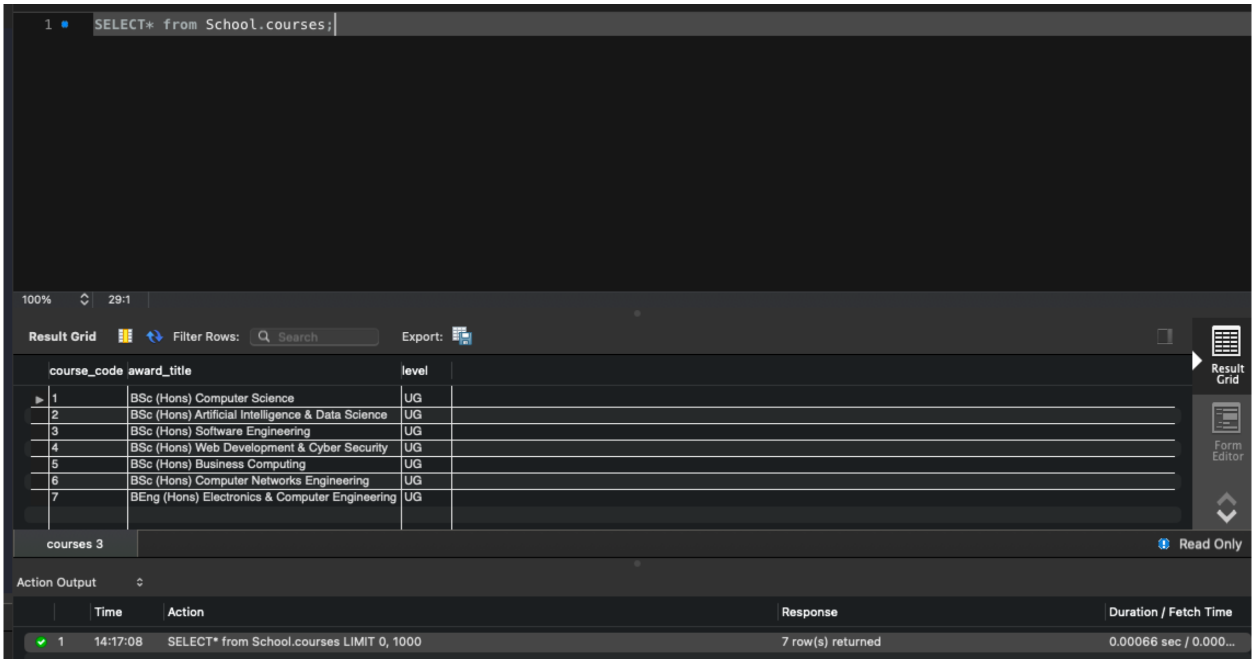


```
SELECT* from School.modules;
```

mod_code	old_code	title	credits	level	semester	pre_requisite	module_leader
CSY1063	CSY1018	Web Development	20	4	S2	none	Chris Rafferty <Chris.Rafferty@northampton.ac....
CSY1064	CSY1019	Software Engineering Fundamentals	20	4	S1	none	Mark Johnson <Mark.Johnson@northampton.a...
CSY1020		Problem Solving & Programming	20	4	S1	none	Mohammed Bahja <Mohammed.Bahja@northa...
CSY1065	CSY1026	Database Fundamentals	20	4	S2	none	Mandy Morrell <Mandy.Morrell@northampton.a...
CSY1030		Digital Footprints	20	4	S2	none	Mandy Morrell <Mandy.Morrell@northampton.a...
CSY1043		Fundamentals of Computing Systems	20	4	S1	none	Michael Opoku Agyeman <Michael.OpokuAgye...
CSY1060		Mathematics for Computer Science	20	4	S2	none	Muawya Eldaw <Muawya.Eldaw@northampton....
ENG1070		Electronic Engineering Practice	20	4	unknown	none	Angel Torres Perez <Angel.TorresPerez@north...

	Time	Action	Response	Duration / Fetch Time
1	14:15:44	SELECT* from School.modules LIMIT 0, 1000	51 row(s) returned	0.0019 sec / 0.00002...

3.5 The "courses" Table:

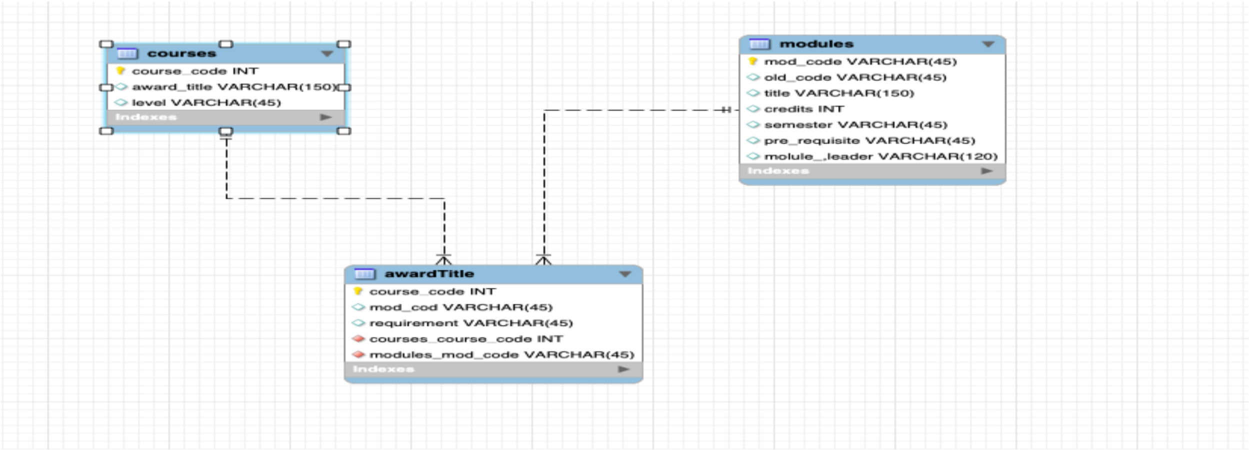


```
SELECT* from School.courses;
```

course_code	award_title	level
1	BSc (Hons) Computer Science	UG
2	BSc (Hons) Artificial Intelligence & Data Science	UG
3	BSc (Hons) Software Engineering	UG
4	BSc (Hons) Web Development & Cyber Security	UG
5	BSc (Hons) Business Computing	UG
6	BSc (Hons) Computer Networks Engineering	UG
7	BEng (Hons) Electronics & Computer Engineering	UG

	Time	Action	Response	Duration / Fetch Time
1	14:17:08	SELECT* from School.courses LIMIT 0, 1000	7 row(s) returned	0.00066 sec / 0.000...

3.6 The picture below shows the ERD diagram for the relational model;

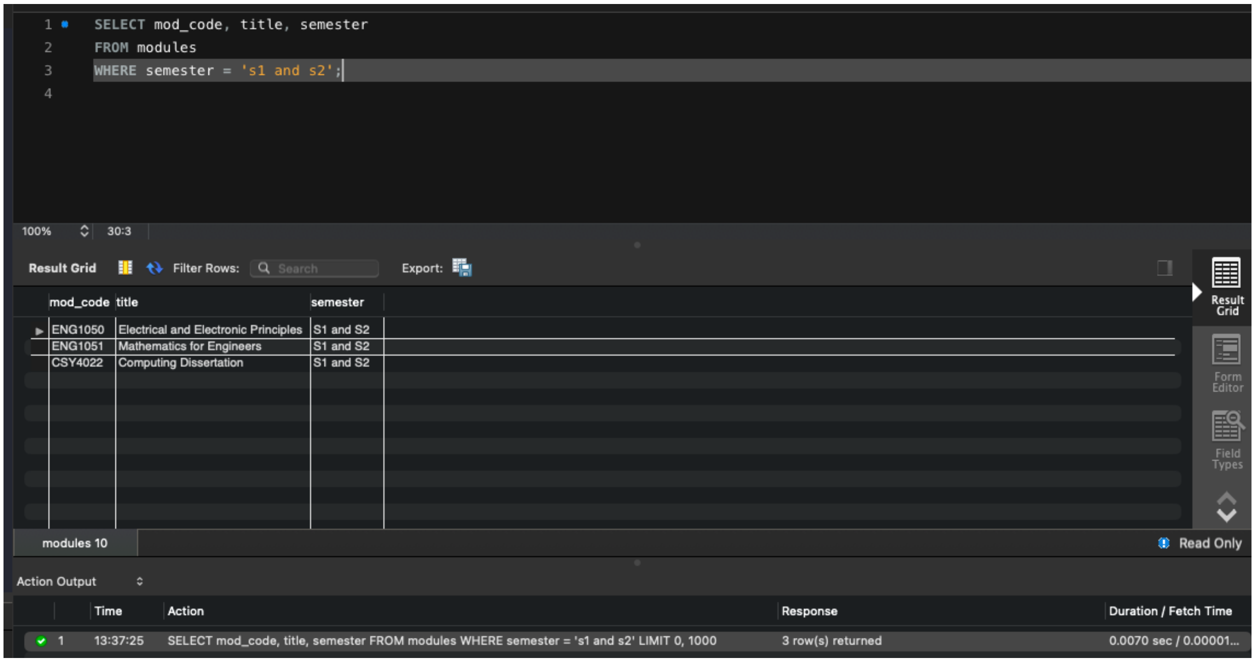


4 Carefully designed SQL and Cypher commands to answer the following queries

4.1 Display those modules running in both semesters

Query

```
SELECT mod_code, title, semester
        FROM modules
        WHERE semester = 's1 and s2';
```



```
SELECT mod_code, title, semester
FROM modules
WHERE semester = 's1 and s2';
```

mod_code	title	semester
ENG1050	Electrical and Electronic Principles	S1 and S2
ENG1051	Mathematics for Engineers	S1 and S2
CSY4022	Computing Dissertation	S1 and S2

	Time	Action	Response	Duration / Fetch Time
✓ 1	13:37:25	SELECT mod_code, title, semester FROM modules WHERE semester = 's1 and s2' LIMIT 0, 1000	3 row(s) returned	0.0070 sec / 0.00001...

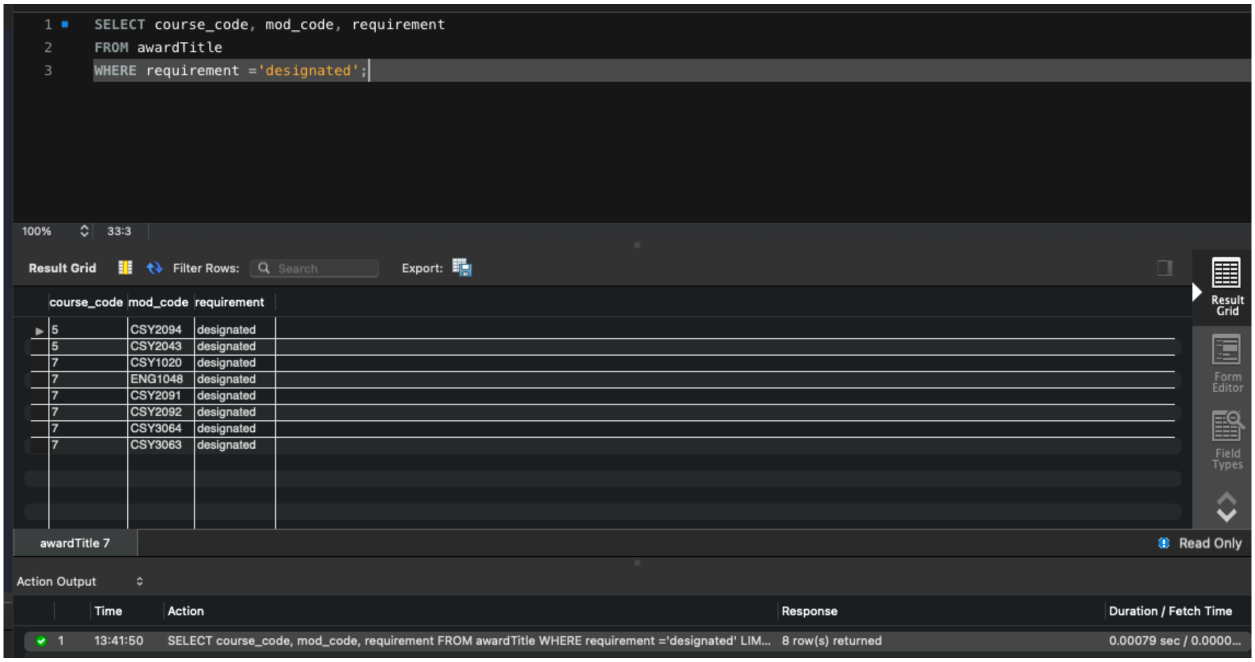
EXPLANATION:

All modules with the semester value of "s1 and s2" will have their module code, title, and semester selected by this query.

4.2 Display all modules with at least a designated module.

Query

```
SELECT course_code, mod_code, requirement
FROM awardTitle
WHERE requirement ='designated';
```



```
SELECT course_code, mod_code, requirement
FROM awardTitle
WHERE requirement ='designated';
```

course_code	mod_code	requirement
5	CSY2094	designated
5	CSY2043	designated
7	CSY1020	designated
7	ENG1048	designated
7	CSY2091	designated
7	CSY2092	designated
7	CSY3064	designated
7	CSY3063	designated

	Time	Action	Response	Duration / Fetch Time
1	13:41:50	SELECT course_code, mod_code, requirement FROM awardTitle WHERE requirement ='designated' LIM...	8 row(s) returned	0.00079 sec / 0.0000...

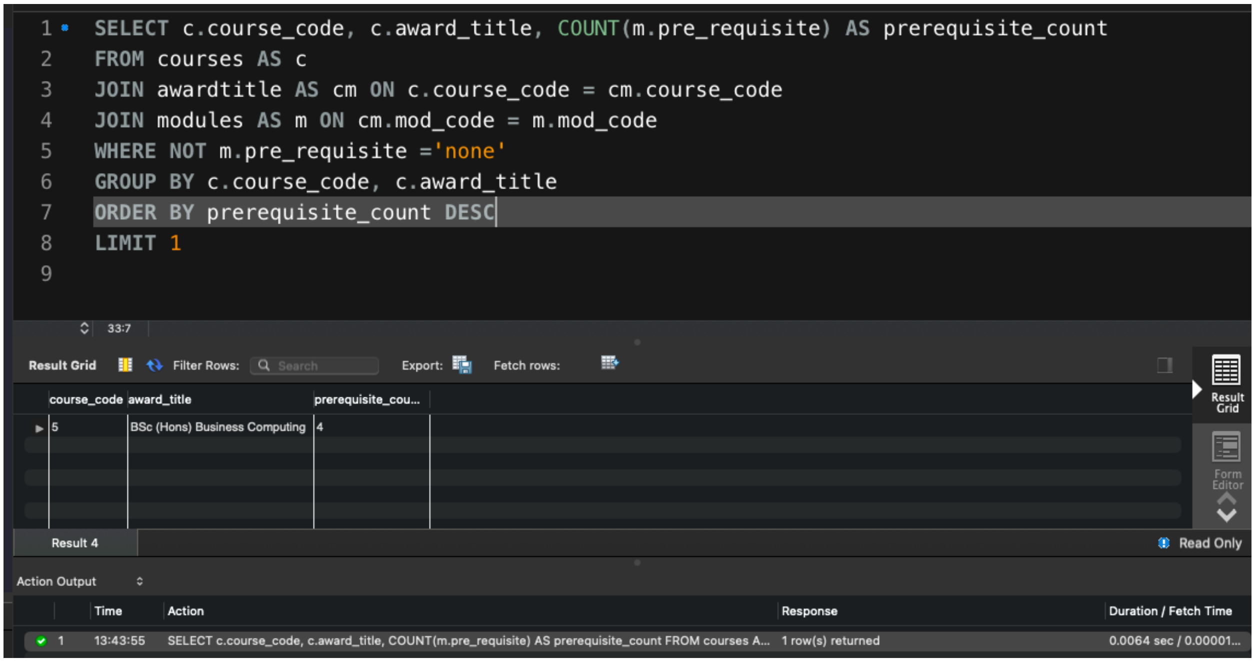
EXPLANATION:

The query retrieves the "course_code," "mod_code," and "requirement" columns from the "awardTitle" table, but only for rows where the "requirement" column is set to 'designated'.

4.3 Which course has the most prerequisites

Query:

```
SELECT c.course_code, c.award_title, COUNT(m.pre_requisite) AS prerequisite_count
FROM courses AS c
JOIN awardtitle AS cm ON c.course_code = cm.course_code
JOIN modules AS m ON cm.mod_code = m.mod_code
WHERE NOT m.pre_requisite ='none'
GROUP BY c.course_code, c.award_title
ORDER BY prerequisite_count DESC
LIMIT 1
```



```
SELECT c.course_code, c.award_title, COUNT(m.pre_requisite) AS prerequisite_count
FROM courses AS c
JOIN awardtitle AS cm ON c.course_code = cm.course_code
JOIN modules AS m ON cm.mod_code = m.mod_code
WHERE NOT m.pre_requisite ='none'
GROUP BY c.course_code, c.award_title
ORDER BY prerequisite_count DESC
LIMIT 1
```

course_code	award_title	prerequisite_cou...
5	BSc (Hons) Business Computing	4

```
    Time      Action                                                                                      Response           Duration / Fetch Time
 1  13:43:55  SELECT c.course_code, c.award_title, COUNT(m.pre_requisite) AS prerequisite_count FROM courses A...  1 row(s) returned  0.0064 sec / 0.00001...
```

Explaination:

This query fetches details about courses such as their codes and award titles. It also calculates the number of prerequisites associated with each course. Finally, it returns the course that has the highest count of prerequisites.

4.4 Display those modules that appear on a single course:

Query:

SELECT m.title, m.mod_code, COUNT(a.course_code) AS COUNT

FROM courses AS c

JOIN awardtitle AS a ON c.course_code = a.course_code

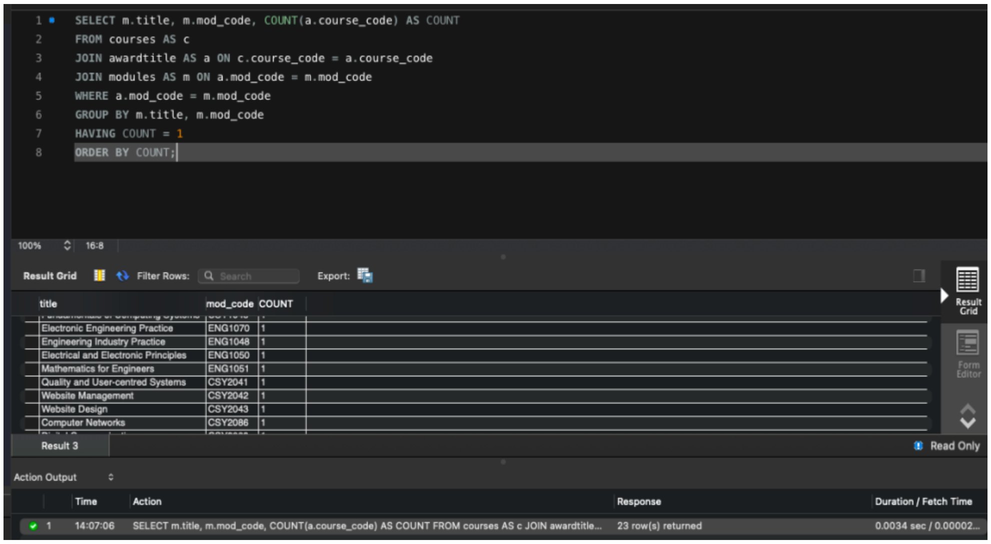
JOIN modules AS m ON a.mod_code = m.mod_code

WHERE a.mod_code = m.mod_code

GROUP BY m.title, m.mod_code

HAVING COUNT = 1

ORDER BY COUNT;



```
SELECT m.title, m.mod_code, COUNT(a.course_code) AS COUNT
FROM courses AS c
JOIN awardtitle AS a ON c.course_code = a.course_code
JOIN modules AS m ON a.mod_code = m.mod_code
WHERE a.mod_code = m.mod_code
GROUP BY m.title, m.mod_code
HAVING COUNT = 1
ORDER BY COUNT;
```

title	mod_code	COUNT
Electronic Engineering Practice	ENG1070	1
Engineering Industry Practice	ENG1048	1
Electrical and Electronic Principles	ENG1050	1
Mathematics for Engineers	ENG1051	1
Quality and User-centred Systems	CSY2041	1
Website Management	CSY2042	1
Website Design	CSY2043	1
Computer Networks	CSY2086	1

	Time	Action	Response	Duration / Fetch Time
1	14:07:06	SELECT m.title, m.mod_code, COUNT(a.course_code) AS COUNT FROM courses AS c JOIN awardtitle...	23 row(s) returned	0.0034 sec / 0.00002...

Explaination:

This query retrieves the "title" and "mod_code" columns from the "modules" table, along with the count of "course_code" from the "awardtitle" table. It performs joins between the tables, filters the results based on matching "mod_code" values, groups the results by "title" and "mod_code", and selects only the groups with a count of 1. The final result is ordered by the count in ascending order.

4.5 Does the computer science course has unblanced work load in both semesters

Query:

SELECT c.course_code, count(c.mod_code) AS Modules,

COUNT(CASE WHEN semester LIKE "S1" THEN "S1" END) AS S1,

COUNT(CASE WHEN semester LIKE "S2" THEN "S2" END) AS S2,

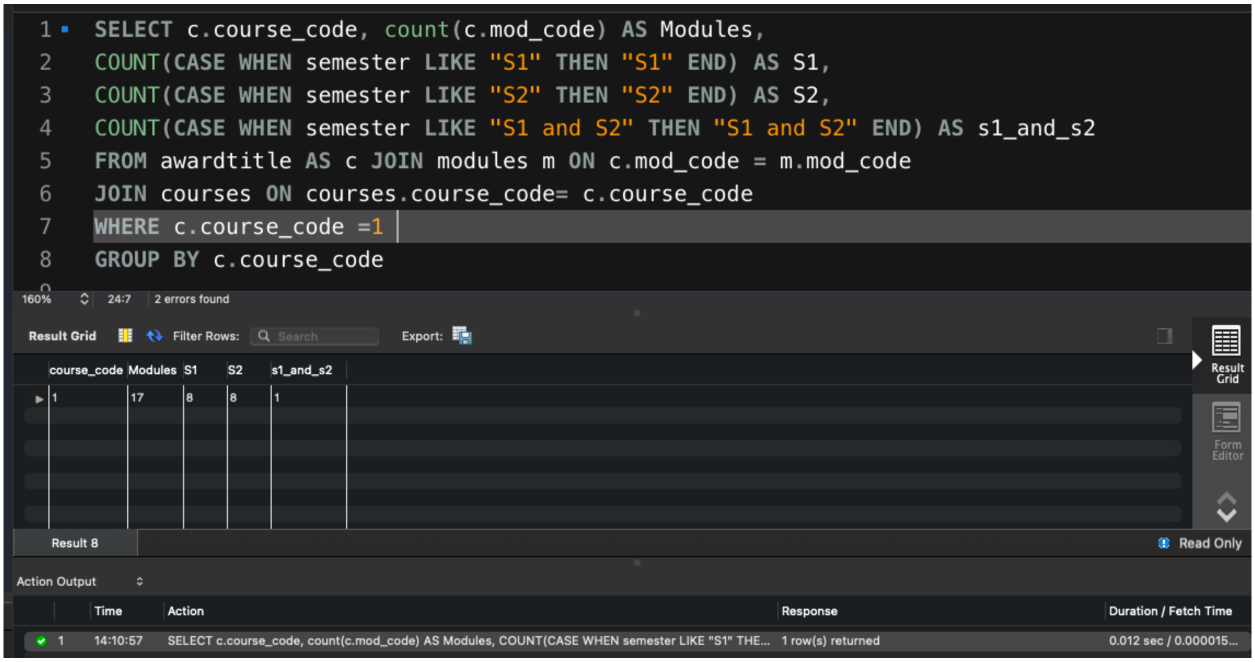
COUNT(CASE WHEN semester LIKE "S1 and S2" THEN "S1 and S2" END) AS s1_and_s2

FROM awardtitle AS c JOIN modules m ON c.mod_code = m.mod_code

JOIN courses ON courses.course_code= c.course_code

WHERE c.course_code =1

GROUP BY c.course_code



```
SELECT c.course_code, count(c.mod_code) AS Modules,
COUNT(CASE WHEN semester LIKE "S1" THEN "S1" END) AS S1,
COUNT(CASE WHEN semester LIKE "S2" THEN "S2" END) AS S2,
COUNT(CASE WHEN semester LIKE "S1 and S2" THEN "S1 and S2" END) AS s1_and_s2
FROM awardtitle AS c JOIN modules m ON c.mod_code = m.mod_code
JOIN courses ON courses.course_code= c.course_code
WHERE c.course_code =1
GROUP BY c.course_code
```

course_code	Modules	S1	S2	s1_and_s2
1	17	8	8	1

	Time	Action	Response	Duration / Fetch Time
1	14:10:57	SELECT c.course_code, count(c.mod_code) AS Modules, COUNT(CASE WHEN semester LIKE "S1" THE...	1 row(s) returned	0.012 sec / 0.000015...

Explaination:

This query fetches the "course_code" column from the "awardtitle" table and calculates the frequency of "mod_code" for each unique "course_code". Additionally, it counts the instances of specific semesters ("S1", "S2", and "S1 and S2") by utilizing the COUNT() function and CASE statements. The query combines multiple tables, applies a filter for a particular "course_code", and groups the output based on the "course_code".

5 Assignment Queries Using Graph Database Model (Neo4j)

The dataset's analysis and the creation procedure:

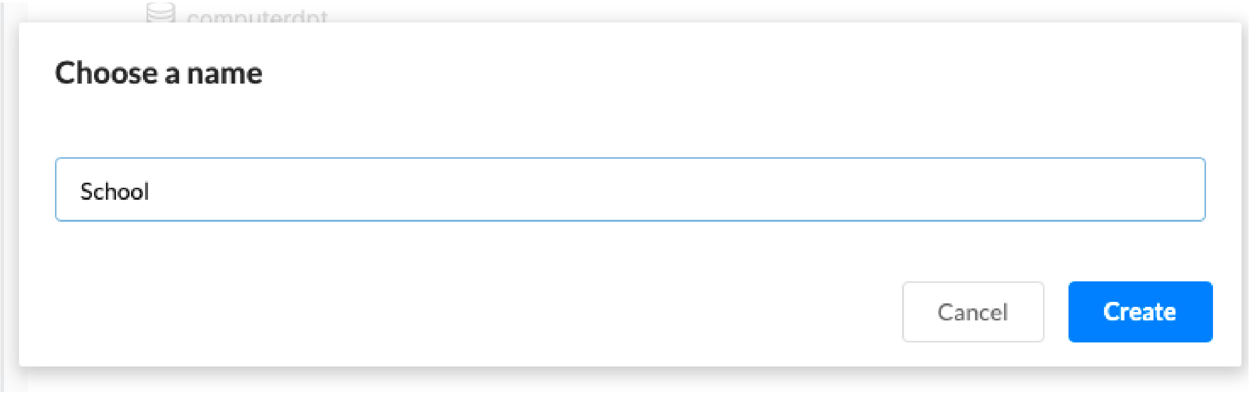
The list of the given nine tables in CSV files includes modules,

I. ai,
II. cs,
III. cse,
IV. ws,
V. courses,

VI. cn,
VII. bc,
VIII. se.

5.1 N eo4j :School

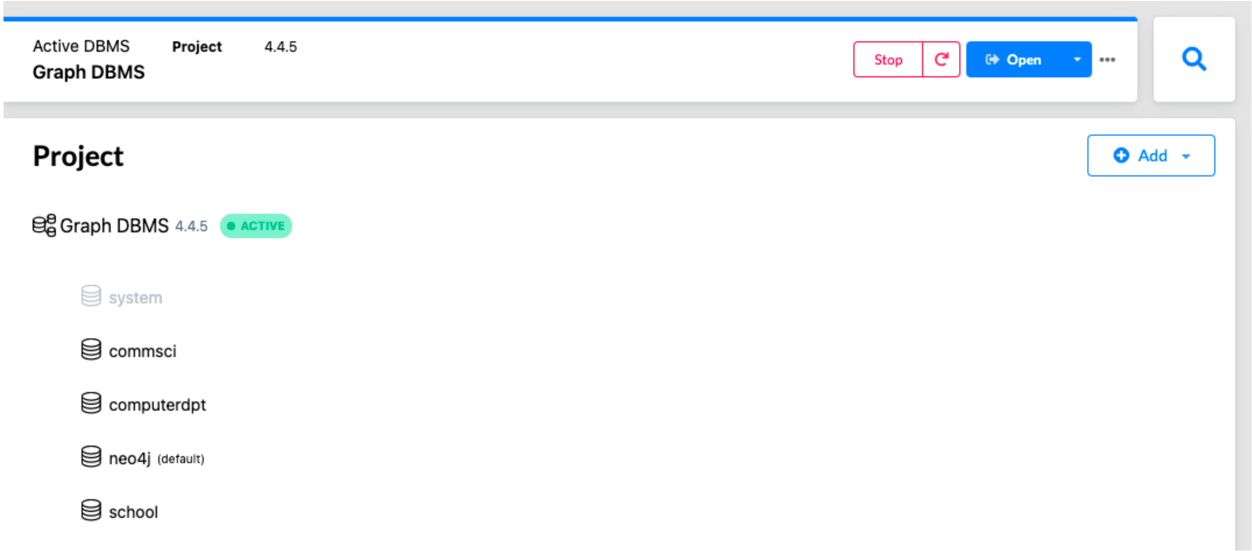
The database was created and named “School” ,the data was cleaned, reformatted and moved to the import folder as shown below:



computerdpt

Choose a name

School

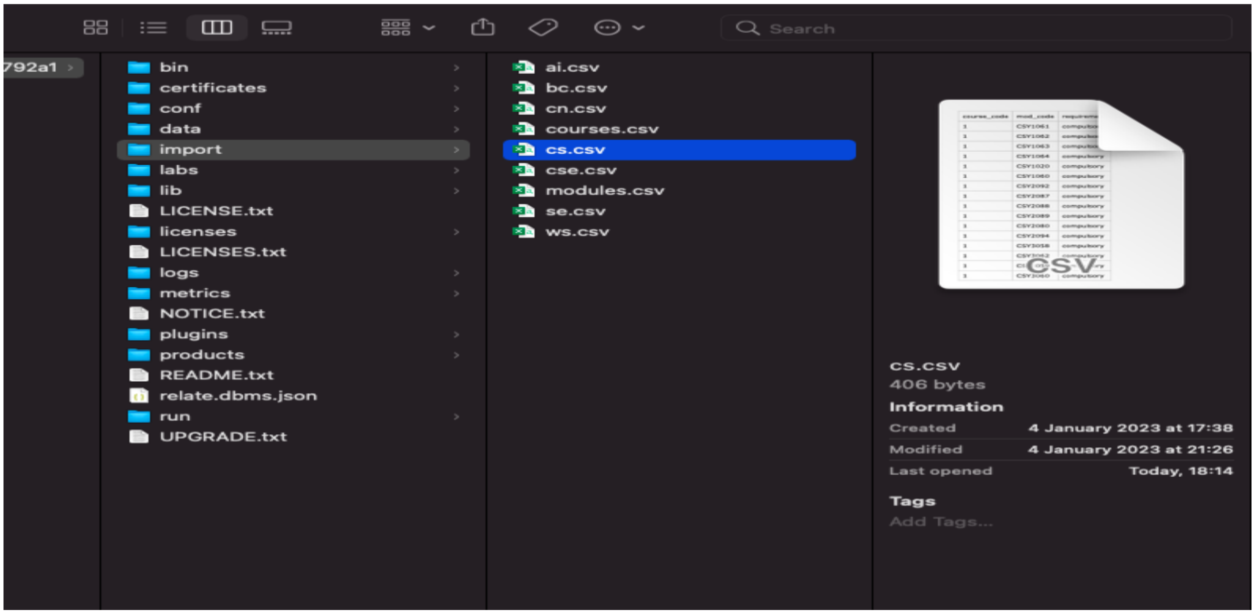


Active DBMS Project 4.4.5
Graph DBMS

Project

Graph DBMS 4.4.5 ● ACTIVE

- system
- commsci
- computerdpt
- neo4j (default)
- school



bin
certificates
conf
data
import
labs
lib
LICENSE.txt
licenses
LICENSES.txt
logs
metrics
NOTICE.txt
plugins
products
README.txt
relate.dbms.json
run
UPGRADE.txt

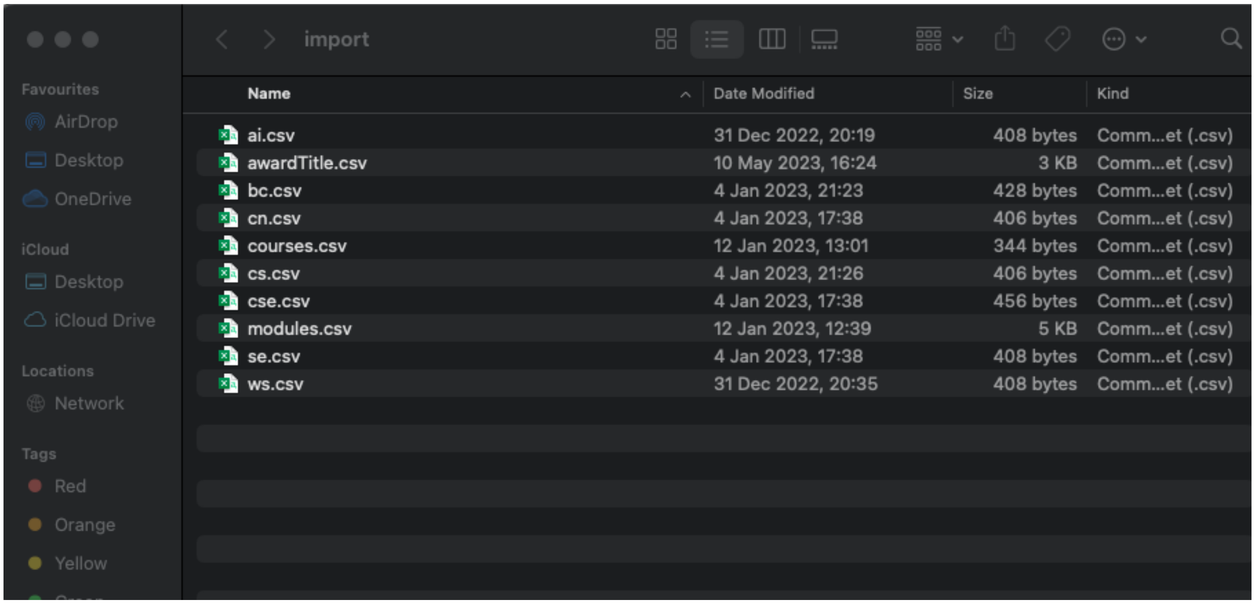
ai.csv
bc.csv
cn.csv
courses.csv
cs.csv
cse.csv
modules.csv
se.csv
ws.csv

cs.csv
406 bytes

Information

Created	4 January 2023 at 17:38
Modified	4 January 2023 at 21:26
Last opened	Today, 18:14

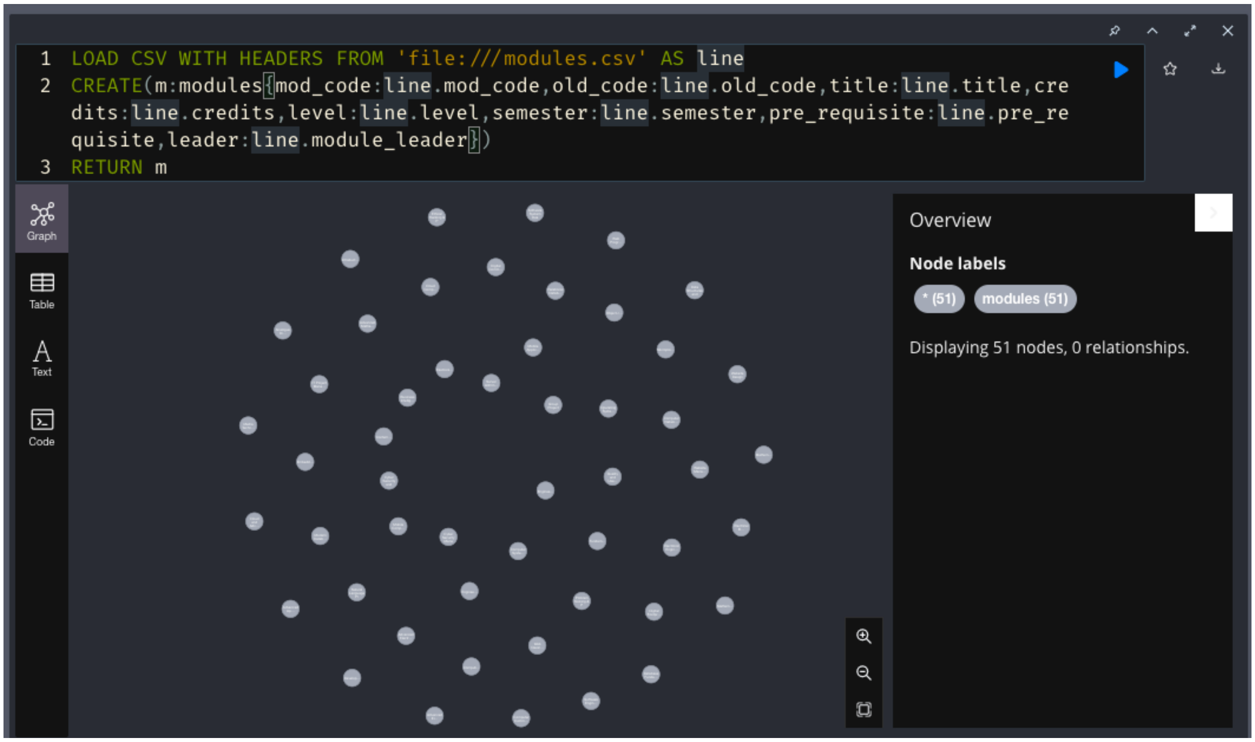
Tags
Add Tags...



Name	Date Modified	Size	Kind
ai.csv	31 Dec 2022, 20:19	408 bytes	Comm...et (.csv)
awardTitle.csv	10 May 2023, 16:24	3 KB	Comm...et (.csv)
bc.csv	4 Jan 2023, 21:23	428 bytes	Comm...et (.csv)
cn.csv	4 Jan 2023, 17:38	406 bytes	Comm...et (.csv)
courses.csv	12 Jan 2023, 13:01	344 bytes	Comm...et (.csv)
cs.csv	4 Jan 2023, 21:26	406 bytes	Comm...et (.csv)
cse.csv	4 Jan 2023, 17:38	456 bytes	Comm...et (.csv)
modules.csv	12 Jan 2023, 12:39	5 KB	Comm...et (.csv)
se.csv	4 Jan 2023, 17:38	408 bytes	Comm...et (.csv)
ws.csv	31 Dec 2022, 20:35	408 bytes	Comm...et (.csv)

5.2 The CSV files were loaded on the school database in no respective order as shown below in the Neo4j browser:

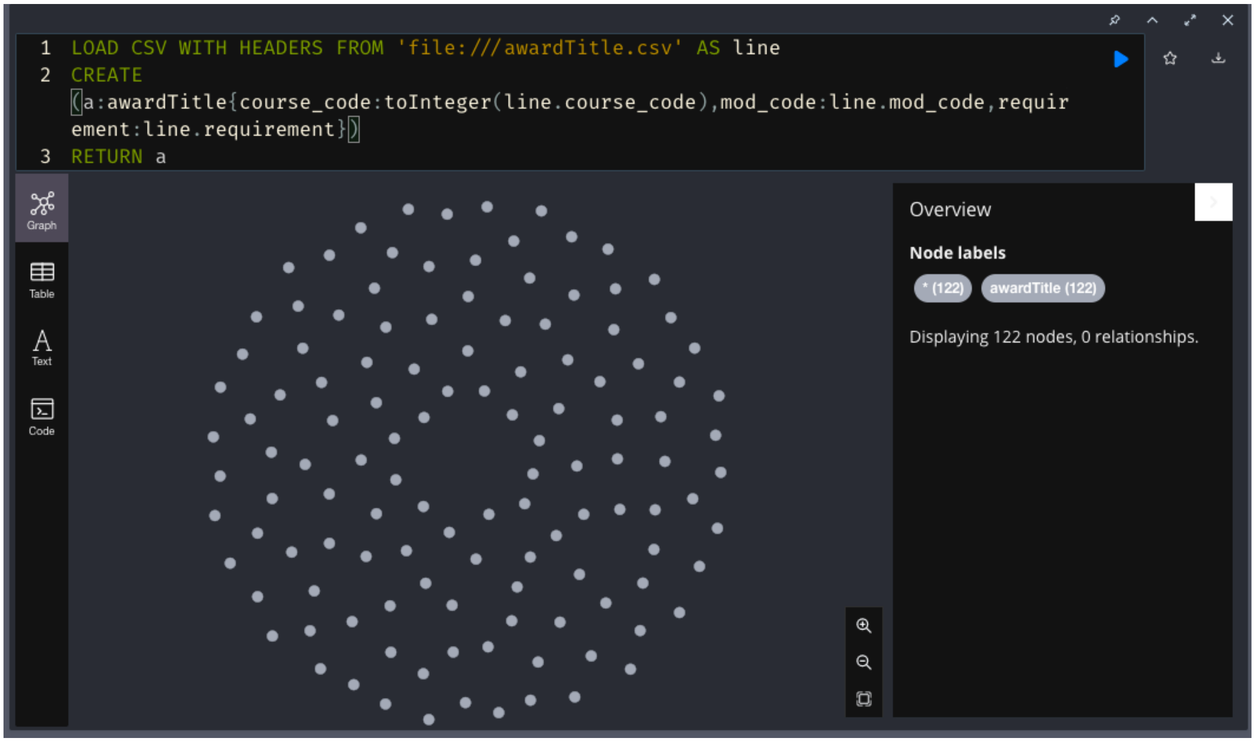
```
LOAD CSV WITH HEADERS FROM 'file:///modules.csv' AS line
CREATE(m:modules{mod_code:line.mod_code,old_code:line.old_code,title:line.title,credits:line.credits,level:line.level,semester:line.semester,pre_requisite:line.pre_requisite,leader:line.module_leader})
RETURN m
```



```
LOAD CSV WITH HEADERS FROM 'file:///modules.csv' AS line
CREATE(m:modules{mod_code:line.mod_code,old_code:line.old_code,title:line.title,credits:line.credits,level:line.level,semester:line.semester,pre_requisite:line.pre_requisite,leader:line.module_leader})
RETURN m
```

Displaying 51 nodes, 0 relationships.

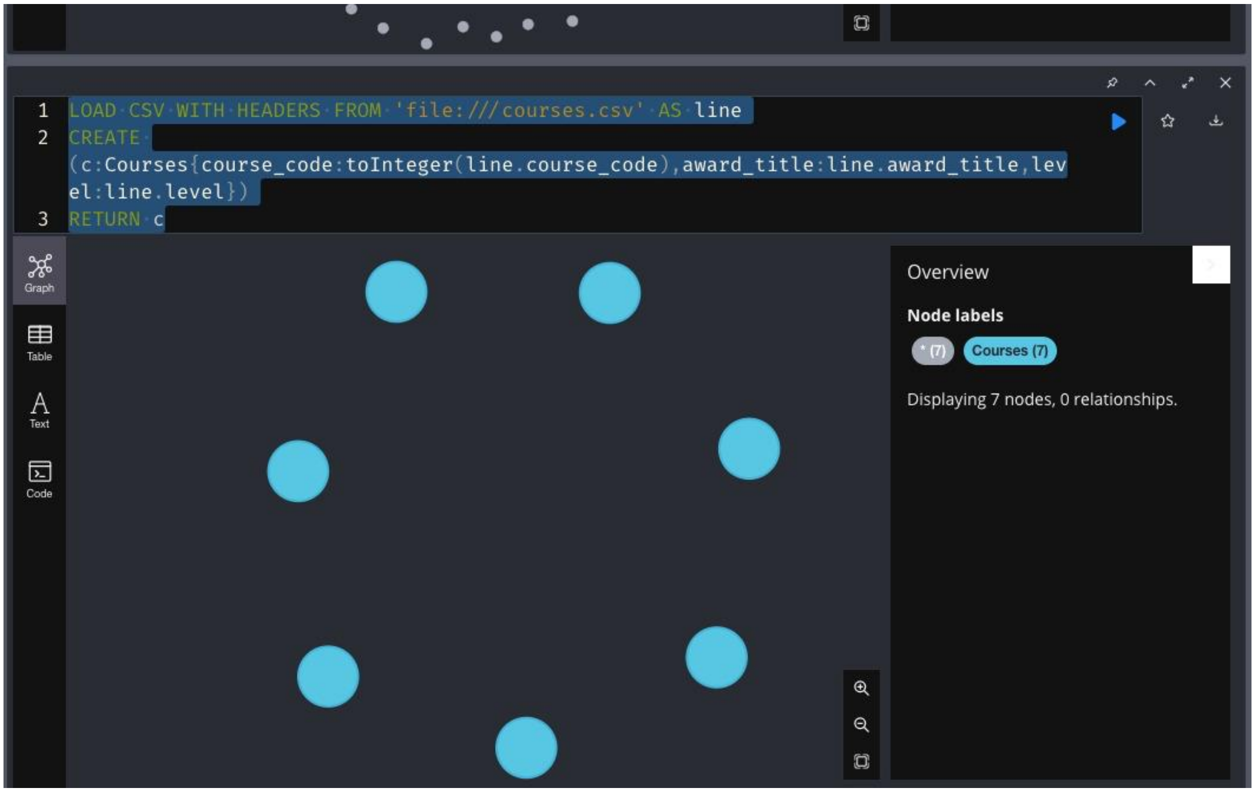
LOAD CSV WITH HEADERS FROM 'file:///awardTitle.csv' AS line
CREATE
(a:awardTitle{course_code:toInteger(line.course_code),mod_code:line.mod_code,requirement:line.requirement})
RETURN a



```
LOAD CSV WITH HEADERS FROM 'file:///awardTitle.csv' AS line
CREATE
(a:awardTitle{course_code:toInteger(line.course_code),mod_code:line.mod_code,requirement:line.requirement})
RETURN a
```

Displaying 122 nodes, 0 relationships.

```
LOAD CSV WITH HEADERS FROM 'file:///courses.csv' AS line
CREATE
(c:Courses{course_code:toInteger(line.course_code),award_title:line.award_title,level:line.level})
RETURN c
```



```
LOAD CSV WITH HEADERS FROM 'file:///courses.csv' AS line
CREATE
(c:Courses{course_code:toInteger(line.course_code),award_title:line.award_title,level:line.level})
RETURN c
```

Overview

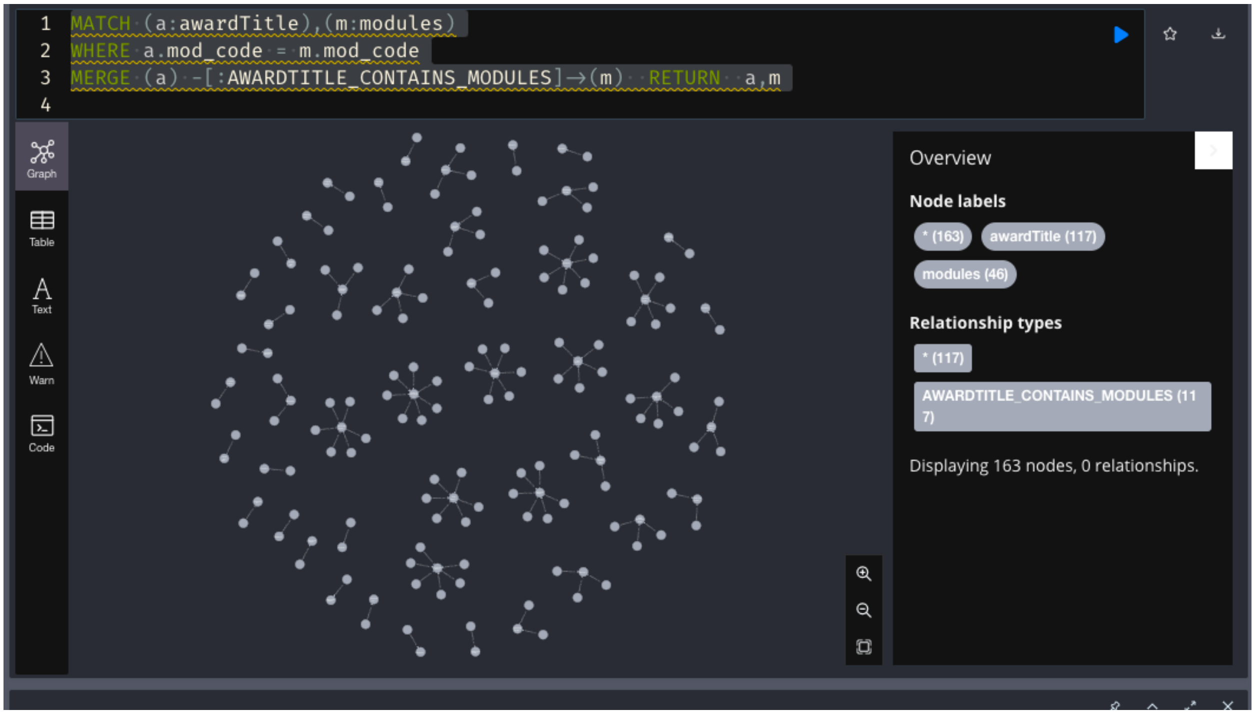
Node labels

* (7) Courses (7)

Displaying 7 nodes, 0 relationships.

5.3 The following Relationships were created as follows:

MATCH (a:awardTitle),(m:modules)
WHERE a.mod_code = m.mod_code
MERGE (a) -[:AWARDTITLE_CONTAINS_MODULES]->(m) RETURN a,m



```
MATCH (a:awardTitle),(m:modules)
WHERE a.mod_code = m.mod_code
MERGE (a) -[:AWARDTITLE_CONTAINS_MODULES]->(m)  RETURN  a,m
```

Overview

Node labels

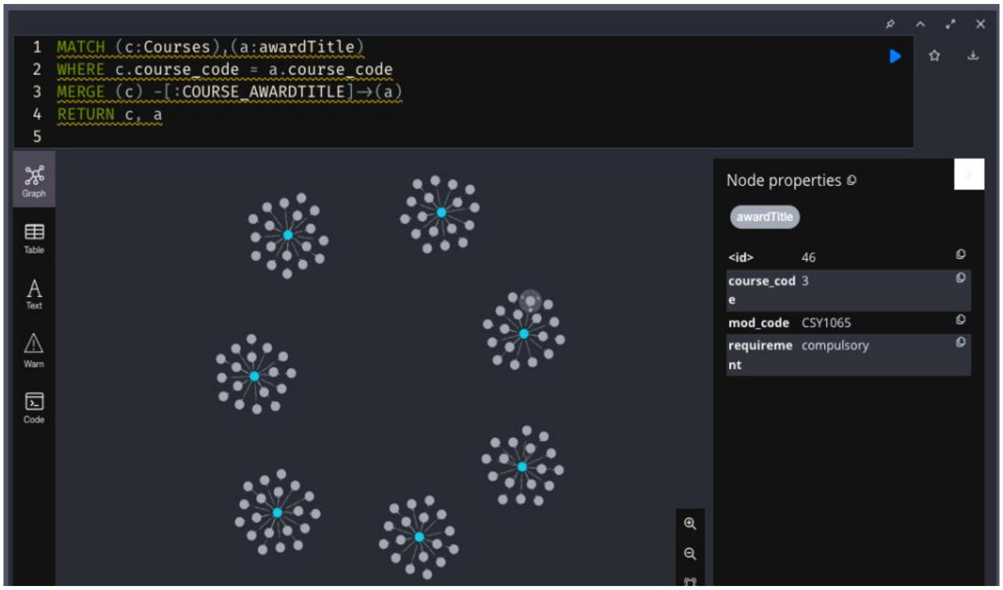
* (163) awardTitle (117)
modules (46)

Relationship types

* (117)
AWARDTITLE_CONTAINS_MODULES (117)

Displaying 163 nodes, 0 relationships.

```
MATCH (c:Courses),(a:awardTitle)
WHERE c.course_code = a.course_code
MERGE (c) -[:COURSE_AWARDTITLE]->(a)
RETURN c, a
```



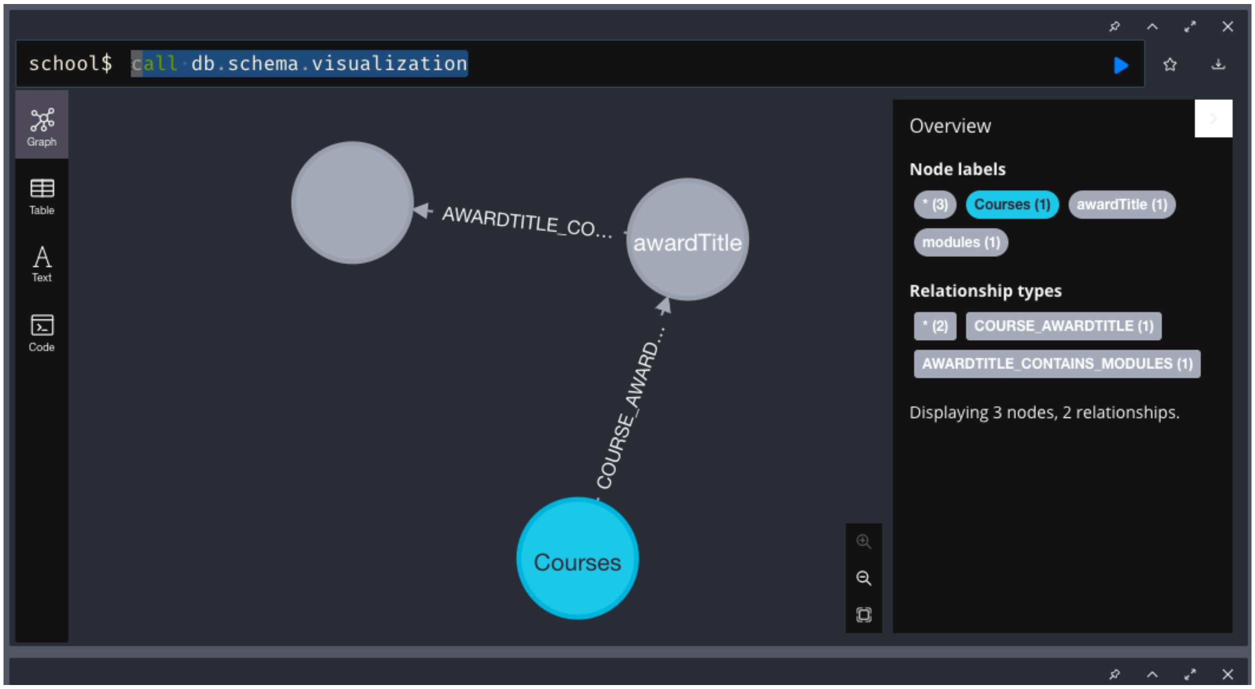
```
MATCH (c:Courses),(a:awardTitle)
WHERE c.course_code = a.course_code
MERGE (c) -[:COURSE_AWARDTITLE]→(a)
RETURN c, a
```

Node properties

awardTitle

<id>	46
course_code	3
mod_code	CSY1065
requirement	compulsory

call db.schema.visualization:



```
school$ call db.schema.visualization
```

Overview

Node labels

* (3) Courses (1) awardTitle (1) modules (1)

Relationship types

* (2) COURSE_AWARDTITLE (1) AWARDTITLE_CONTAINS_MODULES (1)

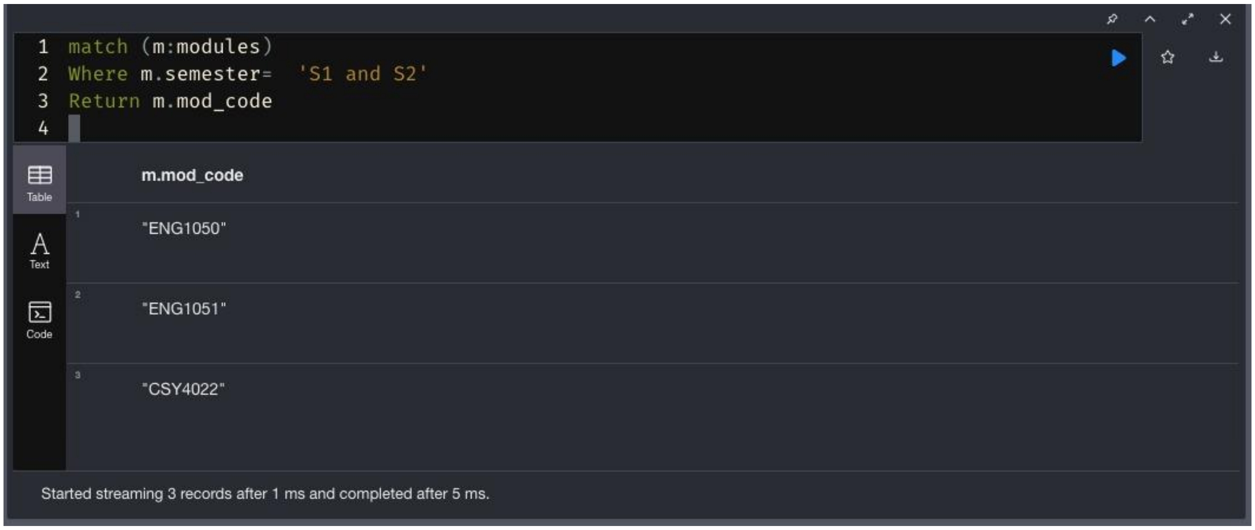
Displaying 3 nodes, 2 relationships.

- Display those modules running in both semesters.
- Display all modules with at least a designated module.
- Which course has most pre-requisites?
- Display those modules that appear on a single course.
- Does the *Computer Science* course have an unbalanced workload in either semester?

6 Neo4j query

6 .1 Display those modules running in both semesters:

match (m:modules)
Where m.semester= 'S1 and S2'
Return m.mod_code



```
match (m:modules)
Where m.semester=  'S1 and S2'
Return m.mod_code
```

```
m.mod_code
1  "ENG1050"
2  "ENG1051"
3  "CSY4022"
```

Started streaming 3 records after 1 ms and completed after 5 ms.

Query Explanation:

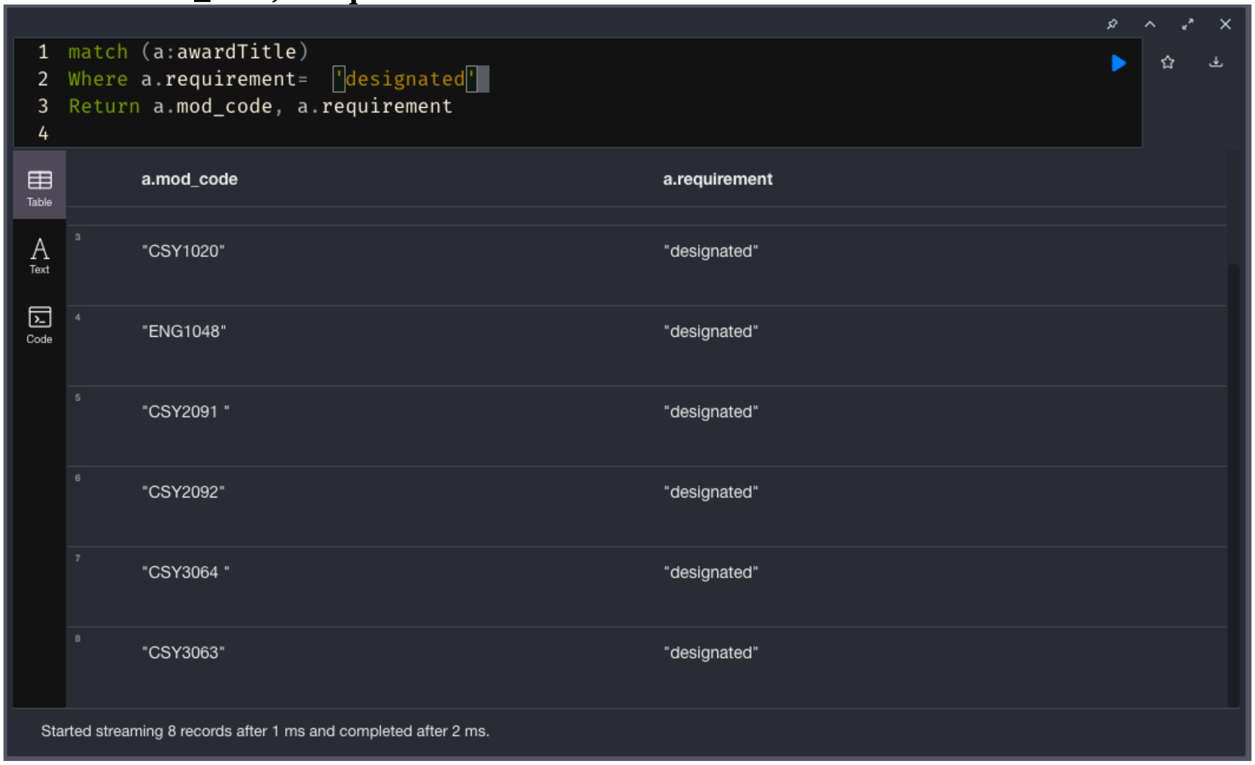
match (m:modules) - this line starts a match clause and defines a node pattern that matches all nodes with the modules label. The nodes are assigned the variable name m.

Where m.semester= 'S1 and S2' - this line filters the nodes based on the value of the semester property. It only returns the nodes where the semester property is equal to the string "S1 and S2".

Return m.mod_code - this line specifies the return value of the query. It returns the mod_code property of the matched nodes, in this case the modules nodes that have a semester property value of "S1 and S2".

6.2 Display all modules with aleast a designated module:

match (a:awardTitle)
Where a.requirement= 'designated'
Return a.mod_code, a.requirement



```
match (a:awardTitle)
Where a.requirement=  'designated'
Return a.mod_code, a.requirement
```

	a.mod_code	a.requirement
3	"CSY1020"	"designated"
4	"ENG1048"	"designated"
5	"CSY2091 "	"designated"
6	"CSY2092"	"designated"
7	"CSY3064 "	"designated"
8	"CSY3063"	"designated"

Started streaming 8 records after 1 ms and completed after 2 ms.

Query Explanation:

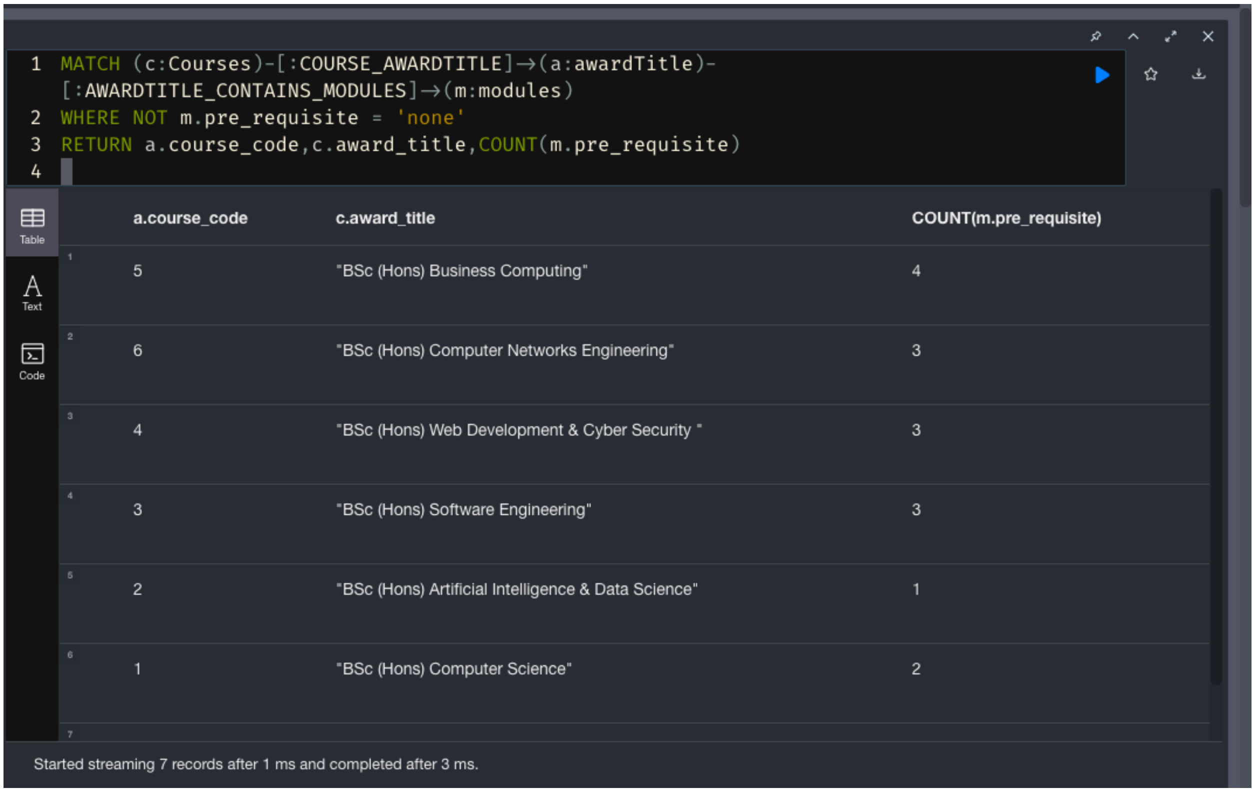
Matching all nodes with the awardTitle label is defined by the node pattern match (a:awardTitle), which begins a match clause. The variable name an is given to the nodes.

This line selects the nodes according to the value of the requirement property when a.requirement='designated'. Only nodes with the requirement attribute equal to the word "designated" are returned.

The query's return value is specified by the line Return a.mod_code, a.requirement. It returns the awardTitle nodes whose requirement property value is "designated" along with their mod_code and requirement properties.

6.3 Which course has most has most pre_requisite

MATCH(c:Courses)-[:COURSE_AWARDTITLE]->(a:awardTitle)-[:AWARDTITLE_CONTAINS_MODULES]->(m:modules)
WHERE NOT m.pre_requisite = 'none'
RETURN a.course_code,c.award_title,COUNT(m.pre_requisite)
ORDER BY COUNT (m.pre_requisite) DESC
LIMIT 1



```
MATCH (c:Courses)-[:COURSE_AWARDTITLE]→(a:awardTitle)-
[:AWARDTITLE_CONTAINS_MODULES]→(m:modules)
WHERE NOT m.pre_requisite = 'none'
RETURN a.course_code,c.award_title,COUNT(m.pre_requisite)
```

	a.course_code	c.award_title	COUNT(m.pre_requisite)
1	5	"BSc (Hons) Business Computing"	4
2	6	"BSc (Hons) Computer Networks Engineering"	3
3	4	"BSc (Hons) Web Development & Cyber Security "	3
4	3	"BSc (Hons) Software Engineering"	3
5	2	"BSc (Hons) Artificial Intelligence & Data Science"	1
6	1	"BSc (Hons) Computer Science"	2
7			

Started streaming 7 records after 1 ms and completed after 3 ms.

Explanation:

The query has a filter condition: WHERE NOT m.pre_requisite = 'none'. The pre_requisite property of the modules node must not be set to the value "none" in order for a path to be included.

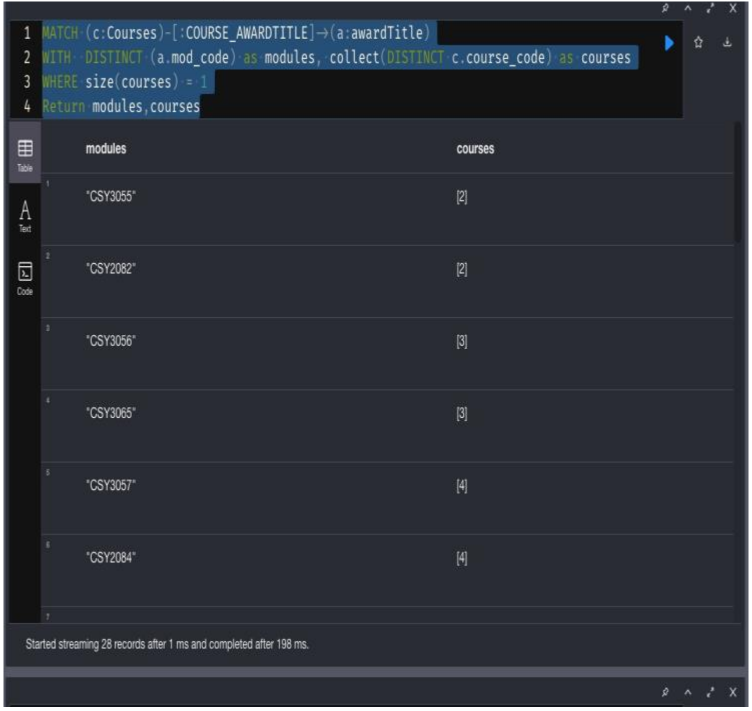
RETURN a.course_code,c.award_title,The query's return value is indicated by the line COUNT(m.pre_requisite). It provides the award's course_code attribute back.The number of times the pre_requisite property does not equal "none" across all of the modules nodes in the matched paths, as well as the award_title and pre_requisite properties of the Courses node's award node.

M.pre_requisite ORDER BY COUNT DESC - This line uses a count of pre_requisite values to sort the results in descending order.

LIMIT 1: This line restricts the results to just the top result, which has the largest number of pre_requisite values among all the modules' nodes.

6.4 Which course appear on a single course:

MATCH (c:Courses)-[:COURSE_AWARDTITLE]->(a:awardTitle)
WITH DISTINCT (a.mod_code) as modules, collect(DISTINCT c.course_code) as courses
WHERE size(courses) = 1
Return modules,courses



```
MATCH (c:Courses)-[:COURSE_AWARDTITLE]→(a:awardTitle)
WITH  DISTINCT (a.mod_code) as modules, collect(DISTINCT c.course_code) as courses
WHERE size(courses) = 1
Return modules,courses
```

modules	courses
"CSY3055"	[2]
"CSY2082"	[2]
"CSY3056"	[3]
"CSY3065"	[3]
"CSY3057"	[4]
"CSY2084"	[4]

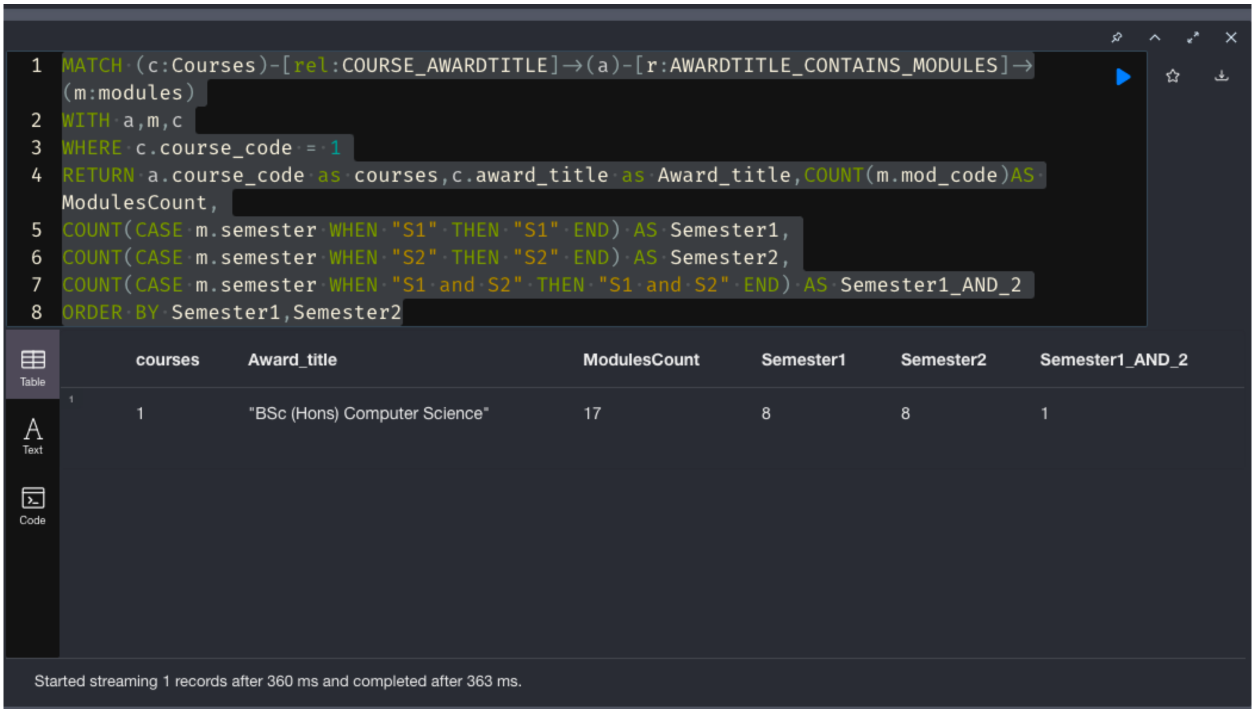
Started streaming 28 records after 1 ms and completed after 198 ms.

Query Explanation:

This query scans the Neo4j database for routes that begin at a node with the Courses label, travel via an outbound relationship of type COURSE_AWARDTITLE, and end at a node with the awardTitle label. The results are then organised by distinct awardTitle node mod_code values, and a list is created by distinct course_code values. Only groups with a single course linked to a certain module are included in the results after filtering. The lists of modules and courses that match the filter condition are returned at the end.

6.5 Does the computer science course has unblanced work load in both semesters

```
        MATCH (c:Courses)-[rel:COURSE_AWARDTITLE]->(a)-
        [r:AWARDTITLE_CONTAINS_MODULES]->(m:modules)

        WITH a,m,c

        WHERE c.course_code = 1

        RETURN a.course_code as courses,c.award_title as
        Award_title,COUNT(m.mod_code)AS ModulesCount,

        COUNT(CASE m.semester WHEN "S1" THEN "S1" END) AS Semester1,

        COUNT(CASE m.semester WHEN "S2" THEN "S2" END) AS Semester2,

        COUNT(CASE m.semester WHEN "S1 and S2" THEN "S1 and S2" END) AS
        Semester1_AND_2

        ORDER BY Semester1,Semester2

    ORDER BY Semester1,Semester2
```



```
MATCH (c:Courses)-[rel:COURSE_AWARDTITLE]→(a)-[r:AWARDTITLE_CONTAINS_MODULES]→(m:modules)
WITH a,m,c
WHERE c.course_code = 1
RETURN a.course_code as courses,c.award_title as Award_title,COUNT(m.mod_code)AS ModulesCount,
COUNT(CASE m.semester WHEN "S1" THEN "S1" END) AS Semester1,
COUNT(CASE m.semester WHEN "S2" THEN "S2" END) AS Semester2,
COUNT(CASE m.semester WHEN "S1 and S2" THEN "S1 and S2" END) AS Semester1_AND_2
ORDER BY Semester1,Semester2
```

	courses	Award_title	ModulesCount	Semester1	Semester2	Semester1_AND_2
1	1	"BSc (Hons) Computer Science"	17	8	8	1

Started streaming 1 records after 360 ms and completed after 363 ms.

Query Explanation:

This search finds all the modules nodes associated with a node with a course_code of 1, counts the number of modules in each semester type, and returns the results. Following that, the query returns the course_code of the n node as computer_science, the total number of mod_code values for the modules nodes as ModulesCount, and the number of semester values for the modules nodes for each semester type. The query then arranges the outcomes according to the Semester1 and Semester2 columns in ascending order.

6.6 Comparison/Reflection of relational and graphical approaches:

The relational database is primarily used for small organisational databases because it is structured in a table and is simple to query. The relational database captures all properties or features entities and makes them unique, but when the data is large and involves several relationships, the relational database cannot capture this efficiently, particularly how one entity relates to another. As a result, non-relational or graph databases are required. This can be observed in databases used by firms such since Facebook, Amazon, and others, as emerging technologies such as Neo4j are particularly efficient with data connections. A look at my assignment above demonstrates how the graph Neo4j was able to capture the relationship latency in SQL queries.