

COMP3008: Set Exercises

Contents

TODO	1
1. Set-exercise 1	1
1.1 Data Model	2
1.2 Database Creation Commands	2
1.3 Design Rationale	3
2. Set-exercise 2	4
3. Set-exercise 3	4
4. Set-exercise 4	4
Appendices	4
Appendix A:	4
Appendix B: 5-Minute Video Demo	4
Appendix C: AI Declaration	5

TODO

- extract Lauren's code and code alike but perhaps better
- relationship in caps
- complex one-shot queries {...}
- LaTeX diagram that has fully context within cw
- remember to utilise the `lecture-notes.md` from ALL the relevant lectures notes, dynamically whilst making the coursework
- Go as far as conceptually as possible, as far complex as the subject goes
- Make the code work ASAP and start talking about it using what's learnt in the module

1. Set-exercise 1

- image/diagram of data model from cw brief
 - add in concise comments to the code somehow
- [X] Create a Neo4j database to store the data comprised in the CSV files. The database should respect the data model displayed in the example in the figure below, please note that the image below has had some nodes hidden for clarity. You have to provide all the commands needed to create the database and populate it with the data in the CSV files, and you must provide them in the exact order you propose to execute them. If you create indexes, you must also include the commands for index creation. Your database will be recreated, and the only way to do so is by following the commands that you will provide, in the order in which you provide them.

The database respects the specified data-model, with **no properties on relationships** as per Lauren's instructions. The intermediate nodes i.e. `Entry` and `Vote`, store the detailed attributes of which would otherwise be properties on relationships.

1.1 Data Model

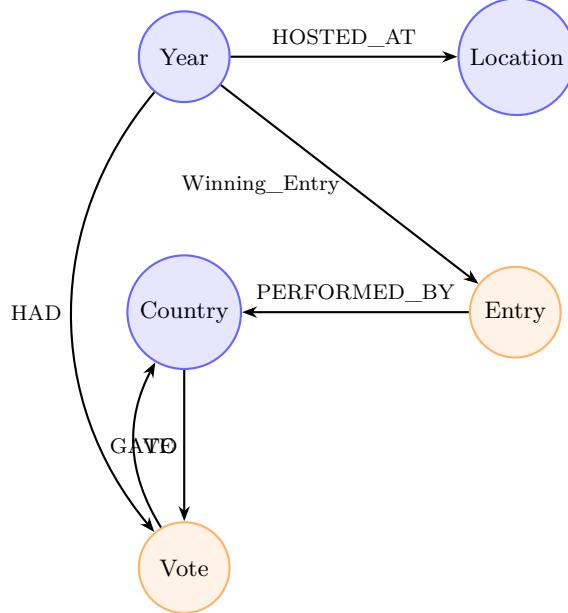


Figure 1: Eurovision Database Data Model (no relationship properties)

Nodes:

Node Label	Properties
<code>Year</code>	<code>year</code> (integer)
<code>Country</code>	<code>name</code> (string, lowercase)
<code>Location</code>	<code>name</code> (string, lowercase)
<code>Entry</code>	<code>song, artist, running_order, total_points</code>
<code>Vote</code>	<code>points, points_type</code>

Relationships (no properties):

Relationship	Pattern
<code>HOSTED_AT</code>	<code>(Year)-[:HOSTED_AT]->(Location)</code>
<code>Winning_Entry</code>	<code>(Year)-[:Winning_Entry]->(Entry)</code>
<code>PERFORMED_BY</code>	<code>(Entry)-[:PERFORMED_BY]->(Country)</code>
<code>GAVE</code>	<code>(Country)-[:GAVE]->(Vote)</code>
<code>TO</code>	<code>(Vote)-[:TO]->(Country)</code>
<code>HAD</code>	<code>(Year)-[:HAD]->(Vote)</code>

1.2 Database Creation Commands

Execute the commands that follow in the exact order shown to recreate the database.

Step 0: Clear existing data (clean slate)

1 `MATCH (n) DETACH DELETE n;`

Step 1: Create uniqueness constraints

```

1 CREATE CONSTRAINT year_unique IF NOT EXISTS
2 FOR (y:Year) REQUIRE y.year IS UNIQUE;
3
4 CREATE CONSTRAINT country_unique IF NOT EXISTS
5 FOR (c:Country) REQUIRE c.name IS UNIQUE;
6
7 CREATE CONSTRAINT location_unique IF NOT EXISTS
8 FOR (l:Location) REQUIRE l.name IS UNIQUE;

```

Step 2: Load Eurovision Winners data

Creates Year, Country, Location, and Entry nodes. Establishes HOSTED_AT, Winning_Entry, and PERFORMED_BY relationships. Note: eurovision_location.csv is not loaded separately as it is a strict subset of this file (see Design Rationale).

```

1 LOAD CSV WITH HEADERS FROM 'file:///Eurovision_Winners.csv' AS row
2 MERGE (y:Year {year: toInteger(row.Year)})
3 MERGE (c:Country {name: toLower(trim(row.Country))})
4 MERGE (l:Location {name: toLower(trim(row.Location))})
5 CREATE (e:Entry {
6   song: row.Song,
7   artist: row.Artist,
8   running_order: toInteger(row.Running_Order),
9   total_points: toInteger(row.Total_Points)
10 })
11 MERGE (y)-[:HOSTED_AT]->(l)
12 CREATE (y)-[:Winning_Entry]->(e)
13 CREATE (e)-[:PERFORMED_BY]->(c);

```

Step 3: Load Voting Results data

Creates Vote nodes and Country nodes for all voting participants. Establishes GAVE, TO, and HAD relationships.

```

1 LOAD CSV WITH HEADERS FROM 'file:///eurovision_results.csv' AS row
2 MERGE (from:Country {name: toLower(trim(row.From))})
3 MERGE (to:Country {name: toLower(trim(row.To))})
4 MERGE (y:Year {year: toInteger(row.Year)})
5 CREATE (v:Vote {
6   points: toInteger(row.Points),
7   points_type: row.Points_type
8 })
9 CREATE (from)-[:GAVE]->(v)
10 CREATE (v)-[:TO]->(to)
11 CREATE (y)-[:HAD]->(v);

```

1.3 Design Rationale

1. **No relationship properties:** Per lecturer requirement, all detailed attributes are stored on intermediate nodes (Entry for winning song details, Vote for voting details) rather than on relationships.
2. **Case normalisation:** Country names are converted to lowercase using `toLower(trim(...))` to ensure consistency between the two CSV files (Winners uses Title Case, Results uses lowercase).
3. **Constraints first:** Uniqueness constraints are created before data loading to ensure data integrity and provide index support for MERGE operations.
4. **MERGE vs CREATE:** MERGE is used for entities that should be unique (Year, Country, Location), whilst CREATE is used for entities where duplicates are expected (Entry, Vote, and their relationships).

5. **eurovision_location.csv not loaded:** This file contains only Year, Country, and Location columns—a strict subset of `Eurovision_Winners.csv`. Both files identically skip 2020 (cancelled due to COVID-19), so no unique Year or Location data exists in the location file. Loading it would create redundant MERGE operations with no effect.

2. Set-exercise 2

3. Set-exercise 3

4. Set-exercise 4

Appendices

Appendix A: ...

Appendix B: 5-Minute Video Demo

- YouTube link: test

Appendix C: AI Declaration

Solo Work	S1 - Generative AI tools have not been used for this assessment.
Assisted Work	A1 – Idea Generation and Problem Exploration Used to generate project ideas, explore different approaches to solving a problem, or suggest features for software or systems. Students must critically assess AI-generated suggestions and ensure their own intellectual contributions are central.
	A2 - Planning & Structuring Projects AI may help outline the structure of reports, documentation and projects. The final structure and implementation must be the student's own work.
	A3 – Code Architecture AI tools maybe used to help outline code architecture (e.g. suggesting class hierarchies or module breakdowns). The final code structure must be the student's own work.
	A4 – Research Assistance Used to locate and summarise relevant articles, academic papers, technical documentation, or online resources (e.g. Stack Overflow, GitHub discussions). The interpretation and integration of research into the assignment remain the student's responsibility.
	A5 - Language Refinement Used to check grammar, refine language, improve sentence structure in documentation not code. AI should be used only to provide suggestions for improvement. Students must ensure that the documentation accurately reflects the code and is technically correct.
	A6 – Code Review AI tools can be used to check comments within the code and to suggest improvements to code readability, structure or syntax. AI should be used only to provide suggestions for improvement. Students must ensure that the code accurately reflects their knowledge and is technically correct.
	A7 - Code Generation for Learning Purposes Used to generate example code snippets to understand syntax, explore alternative implementations, or learn new programming paradigms. Students must not submit AI-generated code as their own and must be able to explain how it works.
	A8 - Technical Guidance & Debugging Support AI tools can be used to explain algorithms, programming concepts, or debugging strategies. Students may also help interpret error messages or suggest possible fixes. However, students must write, test, and debug their own code independently and understand all solutions submitted.
	A9 - Testing and Validation Support AI may assist in generating test cases, validating outputs, or suggesting edge cases for software testing. Students are responsible for designing comprehensive test plans and interpreting test results.
	A10 - Data Analysis and Visualization Guidance AI tools can help suggest ways to analyse datasets or visualize results (e.g. recommending chart types or statistical methods). Students must perform the analysis themselves and understand the implications of the results.
	A11 - Other uses not listed above Please specify:
Partnered Work	P1 - Generative AI tool usage has been used integrally for this assessment Students can adopt approaches that are compliant with instructions in the assessment brief. Please Specify:

Figure 2: Student Declaration of AI Tool use in this Assessment Table

I declare that I've used the AI tools listed below whilst preparing this assessment. I've read and understood the University of Plymouth's policy on the use of AI tools in assessment and confirm that my use falls within

the coursework's allowed categories, i.e. **A2 (Planning and Structuring Projects)** and **A4 (Research Assistance)**.

AI Tool Used	Purpose of Use	Extent of Use
ChatGPT	Finding relevant pages to read in the paper (A4)	Few times if the paper is too long
ChatGPT	General conversations via web-search AI about prevalent papers to read about how the topics relates to others' studies (A4)	Few times at the end

- I understand that the ownership and responsibility for the academic integrity of this submitted assessment falls with me, the student.
- I confirm that all details provide above are an accurate description of how AI was used for this assessment.