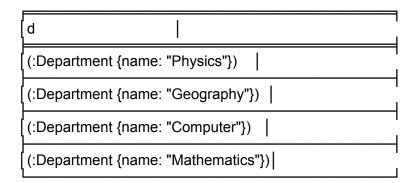
Model the following University information system as a graph model, and answer the following queries using Cypher.

University has various departments like Physics, Geography, Computer etc. Each department conducts various courses and a course may be conducted by multiple departments. Every course may have recommendations provided by people.

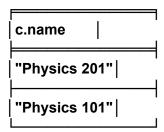
- 1. Identify the labels and relationships, along with their properties, anddraw a high-level Graph model.
- 2. Create nodes and relationships, along with their properties, and visualize your actual Graph model.
- 3. Answer the following Queries:
  - A. List the details of all the departments in the university.
  - B. List the names of the courses provided by Physics department.
  - C. List the most recommended course in Geography department.
  - D. List the names of common courses across Mathematics and computer department.

\_\_\_\_\_\_ CREATE (Physics:Department {name: "Physics"}), (Geography:Department {name: "Geography"}), (Computer:Department {name: "Computer"}), (Mathematics:Department {name: "Mathematics"}), (course1:Course {name: "Physics 101"}), (course2:Course {name: "Physics 201"}), (course3:Course {name: "Geography 101"}), (course4:Course {name: "Computer Science 101"}), (course5:Course {name: "Mathematics 101"}), (person1:Person {name: "John"}), (person2:Person {name: "Alice"}), (Physics)-[:DEPARTMENT CONDUCTS COURSE]->(course1), (Physics)-[:DEPARTMENT\_CONDUCTS\_COURSE]->(course2), (Geography)-[:DEPARTMENT\_CONDUCTS\_COURSE]->(course3), (Computer)-[:DEPARTMENT CONDUCTS COURSE]->(course4), (Mathematics)-[:DEPARTMENT CONDUCTS COURSE]->(course5), (course1)-[:COURSE HAS RECOMMENDATION]->(person1), (course1)-[:COURSE HAS RECOMMENDATION]->(person2), (course2)-[:COURSE\_HAS\_RECOMMENDATION]->(person1), (course3)-[:COURSE HAS RECOMMENDATION]->(person2), (course4)-[:COURSE HAS RECOMMENDATION]->(person1), (course5)-[:COURSE HAS RECOMMENDATION]->(person2);

## Q1)MATCH (d:Department) RETURN d;



2)MATCH (d:Department {name: "Physics"})-[:DEPARTMENT\_CONDUCTS\_COURSE]->(c:Course) RETURN c.name;



## 3)MATCH (d:Department {name:

"Geography"})-[:DEPARTMENT\_CONDUCTS\_COURSE]->(c:Course)-[:COURSE\_HAS\_RECOMM ENDATION]->(p:Person) RETURN c.name, COUNT(p) AS recommendations ORDER BY recommendations DESC LIMIT 1;



4)MATCH (math:Department {name:

"Mathematics"})-[:DEPARTMENT\_CONDUCTS\_COURSE]->(math\_course:Course), (comp:Department {name:

 $"Computer"\})-[:DEPARTMENT\_CONDUCTS\_COURSE]->(comp\_course:Course)$ 

WHERE math\_course.name = comp\_course.name

RETURN math\_course.name;

c.name	recomme	ndations
  "History 101" 3		I

Model the following Library information system as a graph model, and answer the following queries using Cypher.

Consider a library information system having different types of books like text, reference, bibliography etc. A student can buy one or more types of book. A student can recommend or rate a book according to its type.

- 1. Identify the labels and relationships, along with their properties, and draw a high-level Graph model.
- 2. Create nodes and relationships, along with their properties, and visualize your actual Graph model.
- 3. Answer the following Queries:
  - A. List the books of type "text"
  - B. List the name of student who bought a text and reference types books.
  - C. List the most recommended book type.
  - D. List the student who buy the more than one type of book

## CREATE

```
(student1:Student {name: "Alice"}),
(student2:Student {name: "Bob"}),

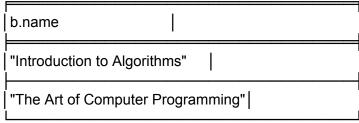
(book1:Book {name: "Introduction to Algorithms"}),
(book2:Book {name: "Database Systems"}),
(book3:Book {name: "The Art of Computer Programming"}),

(type1:Type {name: "Text"}),
(type2:Type {name: "Reference"}),
(type3:Type {name: "Bibliography"}),

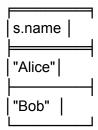
(student1)-[:STUDENT_BOUGHT_BOOK]->(book1)-[:BOOK_OF_TYPE]->(type1),
(student1)-[:STUDENT_BOUGHT_BOOK]->(book2)-[:BOOK_OF_TYPE]->(type2),
(student2)-[:STUDENT_BOUGHT_BOOK]->(book3)-[:BOOK_OF_TYPE]->(type1),

(student1)-[:STUDENT_RECOMMENDED_BOOK]->(book1),
(student1)-[:STUDENT_RECOMMENDED_BOOK]->(book2),
(student2)-[:STUDENT_RECOMMENDED_BOOK]->(book3);
```

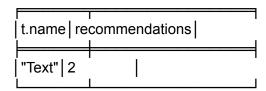
a) MATCH (b:Book)-[:BOOK\_OF\_TYPE]->(t:Type {name: "Text"}) RETURN b.name;



b) MATCH (s:Student)-[:STUDENT\_BOUGHT\_BOOK]->(:Book)-[:BOOK\_OF\_TYPE]->(t:Type) WHERE t.name IN ["Text", "Reference"] RETURN DISTINCT s.name;



c) MATCH
(:Student)-[:STUDENT\_RECOMMENDED\_BOOK]->(b:Book)-[:BOOK\_OF\_TYPE]->(t:Type)
RETURN t.name, COUNT(b) AS recommendations ORDER BY recommendations DESC LIMIT 1;



d) MATCH (s:Student)-[:STUDENT\_BOUGHT\_BOOK]->(b:Book)-[:BOOK\_OF\_TYPE]->(t:Type) WITH s, COUNT(DISTINCT t) AS numTypes WHERE numTypes > 1 RETURN s.name;

