Q1)

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;

d
(:Department {name: "Physics"})
(:Department {name: "Geography"})
(:Department {name: "Computer"})
(:Department {name: "Mathematics"})

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

c.name
"Physics 201"
"Physics 101"

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;

c.name	**recommendations**
"Geography 101"	**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	**recommendations**
"History 101"	**3**

Q2)

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.name
"Introduction to Algorithms"
"The Art of Computer Programming"

b) MATCH (s:Student)-[:STUDENT_BOUGHT_BOOK]->(:Book)-[:BOOK_OF_TYPE]->(t:Type)
WHERE t.name IN ["Text", "Reference"]
RETURN DISTINCT s.name;

s.name
"Alice"
"Bob"

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;

t.name	recommendations
"Text"	2

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;

s.name
"Alice"