# Copilot

Sure, I can help you with that! Let's start by creating the databases and then I'll provide the Cypher queries for each of your requirements.

### 1. Library Database

#### Creating the Database

CREATE DATABASE library;

#### Queries

a) List all people who have issued a book "Book Title":

MATCH (p:Person)-[:ISSUED]->(b:Book {title: "Book Title"})

RETURN p.name;

b) Count the number of people who have read "Book Title":

MATCH (p:Person)-[:READ]->(b:Book {title: "Book Title"})

RETURN COUNT(p);

c) Add a property "Number of books issued" for Mr. Joshi and set its value as the count:

MATCH (p:Person {name: "Mr. Joshi"})-[:ISSUED]->(b:Book)

SET p.numberOfBooksIssued = COUNT(b)

RETURN p;

d) List the names of publishers from Pune city:

MATCH (pub:Publisher {city: "Pune"})

RETURN pub.name;

### 2. Song Database

#### Creating the Database

CREATE DATABASE songs;

#### Queries

a) List the names of songs written by "Author Name":

MATCH (s:Song)-[:WRITTEN\_BY]->(a:Author {name: "Author Name"})

RETURN s.title;

b) List the names of record companies who have financed the song "Song Title":

MATCH (s:Song {title: "Song Title"})-[:FINANCED\_BY]->(rc:RecordCompany)

RETURN rc.name;

c) List the names of artists performing the song "Song Title":

MATCH (s:Song {title: "Song Title"})-[:PERFORMED\_BY]->(a:Artist)

RETURN a.name;

d) Name the songs recorded by the studio "Studio Name":

MATCH (s:Song)-[:RECORDED\_AT]->(studio:Studio {name: "Studio Name"})

RETURN s.title;

### 3. Employee Database

#### Creating the Database

CREATE DATABASE employees;

#### Queries

a) List the names of employees in the department "Department Name":

MATCH (e:Employee)-[:WORKS\_IN]->(d:Department {name: "Department Name"})

RETURN e.name;

b) List the projects along with their properties, controlled by department "Department Name":

MATCH (p:Project)-[:CONTROLLED\_BY]->(d:Department {name: "Department Name"})

RETURN p;

c) List the departments along with the count of employees in it:

MATCH (d:Department)<-[:WORKS\_IN]-(e:Employee)

RETURN d.name, COUNT(e) AS employeeCount;

d) List the skillset for an employee "Employee Name":

MATCH (e:Employee {name: "Employee Name"})-[:HAS\_SKILL]->(s:Skill)

RETURN s.name;

### 4. Movie Database

#### Creating the Database

CREATE DATABASE movies;

#### Queries

a) Find all actors who have acted in a movie "Movie Title":

MATCH (a:Actor)-[:ACTED\_IN]->(m:Movie {title: "Movie Title"})

RETURN a.name;

b) Find all reviewer pairs, one following the other and both reviewing the same movie, and return entire subgraphs:

MATCH (r1:Reviewer)-[:FOLLOWS]->(r2:Reviewer),

(r1)-[:REVIEWED]->(m:Movie),

(r2)-[:REVIEWED]->(m)

RETURN r1, r2, m;

c) Find all actors that acted in a movie together after 2010 and return the actor names and movie node:

MATCH (a1:Actor)-[:ACTED\_IN]->(m:Movie {year: 2010})<-[:ACTED\_IN]-(a2:Actor)

RETURN a1.name, a2.name, m;

d) Find all movies produced by "Producer Name":

MATCH (m:Movie)-[:PRODUCED\_BY]->(p:Producer {name: "Producer Name"})

RETURN m.title;

### 5. Social Network Database

#### Creating the Database

CREATE DATABASE social\_network;

#### Queries

a) Find all friends of "John", along with the year since when John knows them:

MATCH (john:Person {name: "John"})-[:KNOWS {since: year}]->(friend:Person)

RETURN friend.name, year;

b) List out the affiliations of John:

MATCH (john:Person {name: "John"})-[:AFFILIATED\_WITH]->(aff:Affiliation)

RETURN aff.name;

c) Find all friends of John, who are born in the same year as John:

MATCH (john:Person {name: "John"}), (friend:Person)-[:KNOWS]->(john)

WHERE john.birthYear = friend.birthYear

RETURN friend.name;

d) List out the messages posted by John in his timeline during the year 2015:

MATCH (john:Person {name: "John"})-[:POSTED]->(msg:Message)

WHERE msg.year = 2015

RETURN msg.content;

Feel free to ask if you need any further assistance or have more queries!

Source(s)

1. [Create, start, and stop databases - Operations Manual](https://neo4j.com/docs/operations-manual/current/database-administration/standard-databases/create-databases/)

2. [Create and manage a new DBMS locally - Neo4j Desktop](https://neo4j.com/docs/desktop-manual/current/operations/create-dbms/)

3. [How to Set Up a Neo4j Database - Pythian Group](https://www.pythian.com/blog/technical-track/how-to-set-up-a-neo4j-database)

4. [Create Multiple Databases in Community Version](https://community.neo4j.com/t/create-multiple-databases-in-community-version/5025)

5. [Getting started with the Neo4j graph database - SQL Shack](https://www.sqlshack.com/getting-started-with-the-neo4j-graph-database/)