The following questions are related to the graph database system Neo4j [2], and the included Movie database [1].

1. Create three nodes in Cypher to represent three friends – Jack, Jerry and John.

Solution:

```
CREATE
  (jack:User:Male {Name: "Jack"}),
  (jerry:User:Male {Name: "Jerry"}),
  (john:User:Male {Name: "John"})

RETURN
  jack, jerry, john;
```

2. Create relationships showing that:
   - Jack follows Jerry on Twitter.
   - Jack follows John on Twitter.
   - John follows Jerry on Twitter.

Solution:

```
MATCH
  (n:User {Name: "Jack"}), (m:User {Name: "Jerry"})
CREATE
  (n)-[r:FOLLOWS_TWITTER]->(m)
RETURN r;
// OR create two/three at a time:
MATCH
  (n:User {Name: "Jack"}),
  (m:User {Name: "Jerry"}),
  (p:User {Name: "John"})
CREATE
  (n)-[r:FOLLOWS_TWITTER]->(p),
  (p)-[s:FOLLOWS_TWITTER]->(m)
RETURN
  r, s;
```

3. Add some information about Jack, Jerry and John:
   - Jack joined Twitter in May 2010.
   - Jerry joined Twitter in June 2012.
   - John joined Twitter in January 2016.

Solution:

```
MATCH (a:User {Name: "Jack"})
SET a.joinTwitter = "May 2010"
RETURN a;

MATCH
  (a:User {Name: "Jerry"})
  , (b:User {Name: "John"})
SET
  a.joinTwitter = "June 2012"
  , b.joinTwitter = "January 2016"
RETURN
  a, b;
```

4. Add some information about the following relationships:
   - Jack started following Jerry in July 2012.
   - Jack started following John in February 2016.
   - John started following Jerry on January 2016.

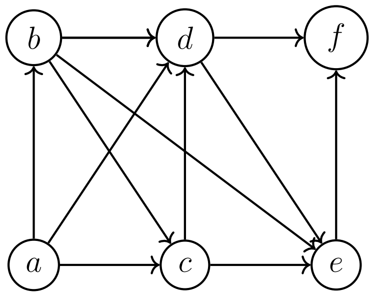
Solution:

```
MATCH
  (a:User {Name: "Jack"})
  -[r:FOLLOWS_TWITTER]->
  (b:User {Name: "Jerry"})
SET
  r.started = "June 2012";

MATCH
  (a:User {Name: "Jack"})
  -[r:FOLLOWS_TWITTER]->
  (b:User {Name: "John"})
SET
  r.started = "February 2016";

MATCH
  (a:User {Name: "John"})
  -[r:FOLLOWS_TWITTER]->
  (b:User {Name: "Jerry"})
SET
  r.started = "January 2016";
```

5. Create the following graph in Cypher:



Solution: Left for the student.

6. The rest of the questions relate to the movie database included with Neo4j. Kevin Bacon is represented by a node in the database. Find the `<id>` of the node.

Solution:

```
MATCH
  (n:Actor)
WHERE
  n.name =~ ".*evin.*acon.*"
RETURN
  ID(n);
```

7. Find a list of the titles of all movies Kevin Bacon has acted in (that are contained in the database).

Solution:

```
MATCH
  (a:Actor)-[:ACTS_IN]->(m:Movie)
WHERE
  ID(a) = 759
RETURN
  m.title;
```

8. Find a list of the titles of all movies Kevin Bacon has acted in, sorted alphabetically.

Solution:

```
MATCH
  (a:Actor)-[:ACTS_IN]->(m:Movie)
WHERE
  ID(a) = 759
RETURN
  m.title
ORDER BY
  m.title;
```

9. Find a list of the titles of all movies Kevin Bacon has acted in, giving only the first five sorted alphabetically.

Solution:

```
MATCH
(a:Actor)-[:ACTS_IN]->(m:Movie)
WHERE
  ID(a) = 759
RETURN
  m.title
ORDER BY
  m.title
LIMIT
  5;
```

10. Find a list of all the names of actors who have acted with Kevin Bacon in a movie, sorted alphabetically.

Solution:

```
MATCH
  (kb:Actor {name: "Kevin Bacon"})
    -[:ACTS_IN]->
  (:Movie)<-[:ACTS_IN]-(a:Actor)
WHERE
  a.name <> "Kevin Bacon"
RETURN
  a.name
ORDER BY
  a.name;
```

11. Find a list of all the names of actors who have acted with an actor who has acted with Kevin Bacon in a movie, sorted alphabetically.

Solution:

```
MATCH
  (kb:Actor {name: "Kevin Bacon"})
    -[:ACTS_IN]->
  (x:Movie)
    <-[:ACTS_IN]-
  (z:Actor)
    -[:ACTS_IN]->
  (y:Movie)
    <-[ACTS_IN]-
  (a:Actor)
RETURN
  DISTINCT a.name
ORDER BY
  a.name;
```

12. Find the three nodes representing Kevin Bacon, Elvis Presley and Edward Asner.

Solution:

```
MATCH
  (n:Actor), (m:Actor), (o:Actor)
WHERE
  n.name =~ ".*lvis.*sley.*" AND
  m.name =~ ".*evin.*acon.*" AND
  o.name =~ ".*ward.*sner.*"
RETURN
  n, m, o;
```

13. Find all the titles movies that Kevin Bacon and Edward Asner have acted in together.

Solution:

```
MATCH
  (n:Actor)-[:ACTS_IN]->(i:Movie)<-[:ACTS_IN]-(m:Actor)
WHERE
  ID(n) = 759
  AND ID(m) = 7767
RETURN
  i.title;
```

14. Find the titles of all movies that Elvis Presley and Edward Asner have acted in together.

Solution:

```
MATCH
  (n:Actor)-[:ACTS_IN]->(i:Movie)<-[:ACTS_IN]-(m:Actor)
WHERE
  ID(n) = 13543
  AND ID(m) = 7767
RETURN
  i.title;
```

15. Find the three nodes representing Kevin Bacon, Elvis Presley and Edward Asner, and all nodes representing any movies that at least two of them have acted in.

Solution:

```
OPTIONAL MATCH
 (n:Actor)-[:ACTS_IN]->(i:Movie)<-[:ACTS_IN]-(m:Actor)
OPTIONAL MATCH
 (n:Actor)-[:ACTS_IN]->(j:Movie)<-[:ACTS_IN]-(o:Actor)
OPTIONAL MATCH
 (m:Actor)-[:ACTS_IN]->(k:Movie)<-[:ACTS_IN]-(o:Actor)
WHERE
  ID(n) = 759
  AND ID(m) = 7767
  AND ID(o) = 13543
RETURN
  n, m, o, i, j, k;
```

16. Show that Elvis Presley has a Bacon number of 2.

Solution:

```
MATCH
  p=shortestPath(
    (kb:Actor {name: "Kevin Bacon"})
    -[r:ACTS_IN*]-
    (ep:Actor {name: "Elvis Presley"})
  )
RETURN
  LENGTH(RELATIONSHIPS(p)) / 2 AS BaconNo;
```

17. Find the names of all actors with a Bacon number of at most 3.

Solution:

```
MATCH
  (kb:Actor {name: "Kevin Bacon"})
  -[r:ACTS_IN*2..6]-
  (x:Actor)
RETURN
  x.name;
```

18. Find the names of all actors with a Bacon number of exactly 3.

Solution:

```
MATCH
  (kb:Actor {name: "Kevin Bacon"})
    -[r:ACTS_IN*6]-
  (x:Actor)
RETURN
  x.name;
```

19. **Extra credit:** Find a distinct list of the types of all the relationships between the *Person* Wes Craven and the *Movie* Scream.

Solution:

```
MATCH
  (n {name: "Wes Craven"})-[r]-(m {title: "Scream"})
RETURN
  TYPE(r)
```

References

[1] neo4j open source graph database.

[2] Neo4j. Neo4j - the world's leading graph database, 2012.