**Question 6 – Cypher querying**

1. *Explain the clauses of the Cypher language and show how it is used for querying.*

**Clauses of the Cypher Language**

MATCH/OPTIONAL MATCH

* **MATCH** finds all paths that satisfy a pattern
* **OPTIONAL MATCH** behaves like a left join, returning **null** when no match is found

WHERE

* Filters results on property conditions or pattern existence (**EXISTS/NOT EXISTS**)

RETURN

* Specifies which variables or expressions to return (like SQL’s **SELECT**)
* Implicit grouping: **RETURN c, COUNT(y)** groups by **c**
* New aggregations: **COLLECT**, etc.

WITH

* Passes intermediate results down the pipeline, enables aliases, and acts as a place to apply further **WHERE, ORDER BY,** or **LIMIT**

UNWIND

* “Explodes” a list into individual rows, useful for processing collections

ORDER BY, SKIP, LIMIT

* Controls result ordering and pagination

SET/DELETE/DETACH DELETE

* **SET** updates properties or labels
* **DELETE** removes relationships or nodes (fails if nodes still have relationships)
* **DETACH DELETE** removes nodes plus all attached relationships

Advanced

* Variable-length paths: **(a)-[:PARENT\*1..5]->(descendant)**
* Shortest path: **MATCH p = shortestPath((a)-[\*]-(b))** (avoid **[\*]** alone)!
* List & pattern comprehensions for inline list generation.

1. *Compare selected queries from your 3 assignments.*

**Comparing of Selected Queries**

“Give me all book titles written by author ID 1.”

**SQL (Assignment 1)**

SELECT book\_name  
FROM book  
WHERE author\_id = 1

**Document (Assignment 2)**

// resolver for GraphQL field **booksByAuthor**

await db.collection('books')

.find({ author\_id: 1 })

.project({ \_id:0, title:1 })

.toArray();

**Graph (Assignment 3)**

query BooksByAuthor($authorId: Int!) {  
 booksByAuthor(authorId: $authorId) {  
 title  
 }  
}

Variables

{  
 "authorId": 1  
}