**Question 7 – MongoDB CRUD operations/neo4j Modifying**

1. *Explain the MongoDB CRUD operations to insert, query, update and delete documents.*

**MongoDB CRUD Operations**

Create (Insert)

* insertOne() used to add a new order

Read (Query)

* findOne() …

Update

* **$set** example
* **$push** example on an embedded array

Delete

* **deleteOne()** used to delete an existing category

Transactions (Advanced)

* wraps multi-document workflows in an ACID transaction

1. *Compare data modification operations in assignment 2 and 3.*

**Comparing Data Modification Document vs. Graph**

**Document (Assignment 2)**

Insert

* **db.orders.insertOne({...})** within an ACID transaction for multi-collection writes

Update

* **db.collection.updateOne( filter, { <operators> } )** e.g. **$set: { title:"New" }, $inc: { "order\_details.0.quantity":1 }, $mul: { price:0.8 }**

Upsert

* **updateOne(filter, {…}, {upsert:true})**

Delete

* **deleteOne** / **deleteMany**, no cascade by default unless via application logic

Transactions

* Multi‐document ACID transactions **(session.startTransaction(), commitTransaction())**

Pattern vs. Separate

* Separate side: do $match stage, then separate $lookup stage

Embedded vs. Separate

* Embedded side: sub-docs (**orders.order\_details**) for atomic writes, references for shared data (authors)

Declarative DSL

* Set of JSON pipelines and CRUD calls

**Graph (Assignment 3)**

Insert

* **CREATE (o:Order {…})** and relationship creation in one statement

Update

* **MATCH (n:Label {…}) SET n.prop = expr [, n.other = expr2 …]** – e.g. **SET b.price = b.price\*0.8, SET r.quantity = r.quantity + 1**

Upsert

* **MERGE** to avoid duplicates

Delete

* **DETACH** **DELETE** to remove a node and all its relationships in one go

Transactions

* Neo4j supports multi-statement ACID [Atomicity, Consistency, Isolation, Durability] transactions in enterprise; single‐query atomic writes by default

Pattern vs. Separate

* Pattern side: express whole traversal from book to author in one declarative pattern
  + Engine does the join for you

Embedded vs. Separate

* Separate side: you model each order-line as a first-class relationship
  + line-item data lives as properties on the **CONTAINS** relationship
  + not embedded in the **Order** node

Declarative DSL

* Every query or mutation is declared once in SDL and executed directly as Cypher, with no extra resolver boilerplate.