Ex No: 7 **Date: 21/10/25**	**Exploring and Querying a Graph Database Using** **Neo4j**

The e-commerce platform ShopSmart uses Neo4j to track users, products, brands, categories, and reviews. Users view, buy, and rate products. Relationships include metadata like purchase dates, quantities, rating values, browsing duration, and similarity scores.

Entities (Nodes):

- Users: 20 (Alice, Bob, Charlie, …)
- Products: 50 (Product1, Product2, …)
- Categories: 10 (Electronics, Books, Clothing, …)
- Brands: 10 (BrandA, BrandB, …)
- Reviews: 10 (Review1, Review2, …)

Relationships:

- :BOUGHT {date, quantity, price_paid}, :VIEWED {date, duration_seconds, device}, :RATED {rating, review_date}, :REVIEWS {review_id}, :BELONGS_TO {added_date}, :MADE_BY {launch_year}, :SIMILAR_TO {similarity_score}, :FRIENDS_WITH {since, interaction_count}

CREATE Questions (with Properties)

1. Create user nodes: Alice, Bob, Charlie with emails.
2. Create product nodes: Product1 (Electronics, 49.99), Product2 (Books, 29.99), Product3 (Clothing, 39.99).
3. Create category nodes: Electronics, Books, Clothing.
4. Create brand nodes: BrandA, BrandB, BrandC.
5. Create review nodes: Review1 (rating 5), Review2 (rating 4), Review3 (rating 3).
6. Create BOUGHT relationship: Alice bought Product1 on 2025-10-14, quantity 2, price_paid 99.98.
7. Create BOUGHT relationship: Bob bought Product2 on 2025-10-13, quantity 1, price_paid 29.99.
8. Create VIEWED relationship: Charlie viewed Product3 on 2025-10-12 for 120 seconds on mobile.
9. Create BELONGS_TO relationship: Product1 belongs to Electronics, added_date 2025-01-01.
10. Create MADE_BY relationship: Product1 made by BrandA, launch_year 2024.
11. Create RATED relationship: Alice rated Review1 on 2025-10-14; Review1 reviews Product1.
12. Create SIMILAR_TO relationship: Product1 is similar to Product3, similarity_score 0.85.
13. Create FRIENDS_WITH relationship: Alice friends with Bob since 2023-01-01, interaction_count 15.

14. Create BOUGHT relationship: Charlie bought Product2 on 2025-10-14, quantity 1, price_paid 29.99.
15. Create VIEWED relationship: Alice viewed Product2 on 2025-10-13 for 45 seconds on desktop.

QUERY Questions (with Properties)

16. Find all products purchased by Alice, including quantity and date.
17. Recommend products for Alice based on products Bob bought in the last month.
18. List all products in the Electronics category and show when they were added.
19. Find top 5 products by average rating, including the latest review date.
20. Find all products made by BrandB and launch year.
21. Find users who viewed Product3 but did not buy it; show viewing duration and device.
22. Find the category with the highest total purchased quantity.
23. Find all users who rated Product1 with 5 stars, including review date.
24. Suggest products similar to Product1 with similarity_score > 0.8.
25. Find friends of Alice who purchased Product2, showing purchase date.
26. List all reviews for Product2 along with user and rating.
27. Find users who bought products from multiple categories, showing product names and categories.
28. Find products viewed more than 100 seconds by any user.
29. Recommend products based on friends' purchases in the last 30 days.
30. Find products frequently bought together with Product2, including purchase dates.

UPDATE Questions (with Properties)

31. Update the price of Product1 to 99.99.
32. Add Product1 to a new category "Gadgets" with added_date 2025-10-14.
33. Update Alice's email to alice_new@example.com.
34. Update Review1 rating to 4 on 2025-10-15.
35. Add a BOUGHT relationship: Charlie bought Product3 on 2025-10-14, quantity 1, price_paid 39.99.
36. Change the brand of Product3 from BrandC to BrandD, launch_year 2025.
37. Rename Product2 to "BookMaster 2025".
38. Update the names of users Bob and Charlie to Robert and Charles.
39. Change category of Product3 from Clothing to Sports, added_date 2025-10-14.
40. Add a new attribute discount = 10% to Product1.

DELETE Questions (with Properties)

41. Delete Product50 from the database.
42. Delete Review5 with rating 2.
43. Remove BOUGHT relationship between Alice and Product2.

44. Delete category “OldCategory” and all associated relationships.
45. Delete user Tina and all relationships.
46. Remove SIMILAR_TO relationship between Product1 and Product3.
47. Delete all products made by BrandJ.
48. Delete all VIEWED relationships for Product3 before 2025-01-01.
49. Delete products never bought or reviewed (Product48, Product49).
50. Delete Review3 without deleting Product3.

ANALYTICAL / COMPLEX Query Questions (with Properties)

51. Find top 5 users by purchase quantity in October 2025.
52. Recommend products for Alice based on purchases in Electronics category.
53. Identify products frequently bought together with Product2 in the last 30 days.
54. Find average rating for products made by BrandA.
55. Suggest products for Alice based on her friends’ purchases, including purchase dates.
56. Find users who bought products with price_paid > 80.
57. Identify categories generating the highest total revenue (sum of price_paid × quantity).
58. Find products viewed >100 seconds but never bought.
59. List products with review count and average rating.
60. Identify potential bundles of Product1, Product2, Product3 based on co-purchases.

Query:

```
// 1–5: Create Users, Products, Categories, Brands, Reviews

CREATE

(:User {name: 'Alice', email: 'alice@example.com'}),

(:User {name: 'Bob', email: 'bob@example.com'}),

(:User {name: 'Charlie', email: 'charlie@example.com'}),

(:Product {name: 'Product1', category: 'Electronics', price: 49.99}),

(:Product {name: 'Product2', category: 'Books', price: 29.99}),

(:Product {name: 'Product3', category: 'Clothing', price: 39.99}),

(:Category {name: 'Electronics'}),

(:Category {name: 'Books'}),

(:Category {name: 'Clothing'}),

(:Brand {name: 'BrandA'}),
(:Brand {name: 'BrandB'}),

(:Brand {name: 'BrandC'}),

(:Review {name: 'Review1', rating: 5}),

(:Review {name: 'Review2', rating: 4}),

(:Review {name: 'Review3', rating: 3});

// 6

MATCH (a:User {name:'Alice'}), (p:Product {name:'Product1'})

CREATE (a)-[:BOUGHT {date: date('2025-10-14'), quantity: 2, price_paid: 99.98}]->(p);

// 7

MATCH (b:User {name:'Bob'}), (p:Product {name:'Product2'})

CREATE (b)-[:BOUGHT {date: date('2025-10-13'), quantity: 1, price_paid: 29.99}]-
>(p);

// 8

MATCH (c:User {name:'Charlie'}), (p:Product {name:'Product3'})

CREATE (c)-[:VIEWED {date: date('2025-10-12'), duration_seconds: 120, device:
'mobile'}]->(p);

// 9

MATCH (p:Product {name:'Product1'}), (c:Category {name:'Electronics'})

CREATE (p)-[:BELONGS_TO {added_date: date('2025-01-01')}]->(c);

// 10

MATCH (p:Product {name:'Product1'}), (b:Brand {name:'BrandA'})

CREATE (p)-[:MADE_BY {launch_year: 2024}]->(b);

// 11

MATCH (a:User {name:'Alice'}), (r:Review {name:'Review1'}), (p:Product
{name:'Product1'})

CREATE (a)-[:RATED {date: date('2025-10-14')}]->(r),

    (r)-[:REVIEWS]->(p);
```

```
// 12

MATCH (p1:Product {name:'Product1'}), (p3:Product {name:'Product3'})

CREATE (p1)-[:SIMILAR_TO {similarity_score: 0.85}]->(p3);

// 13

MATCH (a:User {name:'Alice'}), (b:User {name:'Bob'})

CREATE (a)-[:FRIENDS_WITH {since: date('2023-01-01'), interaction_count: 15}]-
>(b);

// 14

MATCH (c:User {name:'Charlie'}), (p:Product {name:'Product2'})

CREATE (c)-[:BOUGHT {date: date('2025-10-14'), quantity: 1, price_paid: 29.99}]->(p);

// 15

MATCH (a:User {name:'Alice'}), (p:Product {name:'Product2'})

CREATE (a)-[:VIEWED {date: date('2025-10-13'), duration_seconds: 45, device:
'desktop'}]->(p);

// 16

MATCH (a:User {name:'Alice'})-[b:BOUGHT]->(p:Product)

RETURN p.name AS Product, b.quantity AS Quantity, b.date AS PurchaseDate;

// 17

MATCH (a:User {name:'Alice'})-[:FRIENDS_WITH]->(b:User)-[bo:BOUGHT]-
>(p:Product)

WHERE bo.date >= date('2025-09-14')

AND NOT (a)-[:BOUGHT]->(p)

RETURN p.name AS RecommendedProduct, b.name AS Friend;

// 18

MATCH (p:Product)-[r:BELONGS_TO]->(c:Category {name:'Electronics'})

RETURN p.name AS Product, r.added_date AS AddedDate;
```

```
// 19

MATCH (r:Review)-[:REVIEWS]->(p:Product)

RETURN p.name AS Product, AVG(r.rating) AS AvgRating, MAX(r.date) AS
LatestReviewDate

ORDER BY AvgRating DESC LIMIT 5;

// 20

MATCH (p:Product)-[m:MADE_BY]->(b:Brand {name:'BrandB'})

RETURN p.name AS Product, m.launch_year AS LaunchYear;

// 21

MATCH (u:User)-[v:VIEWED]->(p:Product {name:'Product3'})

WHERE NOT (u)-[:BOUGHT]->(p)

RETURN u.name AS User, v.duration_seconds AS Duration, v.device AS Device;

// 22

MATCH (u:User)-[b:BOUGHT]->(p:Product)-[:BELONGS_TO]->(c:Category)

RETURN c.name AS Category, SUM(b.quantity) AS TotalQuantity

ORDER BY TotalQuantity DESC LIMIT 1;

// 23

MATCH (u:User)-[r:RATED]->(rev:Review {rating:5})-[:REVIEWS]->(p:Product
{name:'Product1'})

RETURN u.name AS User, r.date AS ReviewDate;

// 24

MATCH (p1:Product {name:'Product1'})-[s:SIMILAR_TO]->(p2:Product)

WHERE s.similarity_score > 0.8

RETURN p2.name AS SimilarProduct, s.similarity_score AS Score;

// 25

MATCH (a:User {name:'Alice'})-[:FRIENDS_WITH]->(f:User)-[b:BOUGHT]-
>(p:Product {name:'Product2'})
```

```
RETURN f.name AS Friend, b.date AS PurchaseDate;

// 26

MATCH (u:User)-[:RATED]->(r:Review)-[:REVIEWS]->(p:Product {name:'Product2'})

RETURN u.name AS Reviewer, r.rating AS Rating;

// 27

MATCH (u:User)-[:BOUGHT]->(p:Product)-[:BELONGS_TO]->(c:Category)

WITH u, COLLECT(DISTINCT c.name) AS Categories, COLLECT(p.name) AS
Products

WHERE SIZE(Categories) > 1

RETURN u.name AS User, Products, Categories;

// 28

MATCH (u:User)-[v:VIEWED]->(p:Product)

WHERE v.duration_seconds > 100

RETURN u.name AS User, p.name AS Product, v.duration_seconds AS Duration;

// 29

MATCH (a:User {name:'Alice'})-[:FRIENDS_WITH]->(f:User)-[b:BOUGHT]-
>(p:Product)

WHERE b.date >= date('2025-09-14')

AND NOT (a)-[:BOUGHT]->(p)

RETURN f.name AS Friend, p.name AS RecommendedProduct;

// 30

MATCH (u:User)-[b1:BOUGHT]->(p1:Product {name:'Product2'})

MATCH (u)-[b2:BOUGHT]->(p2:Product)

WHERE p2 <> p1

RETURN p2.name AS CoPurchasedProduct, b1.date AS PurchaseDate;

// 31
```

```
MATCH (p:Product {name:'Product1'})

SET p.price = 99.99;

// 32

MATCH (p:Product {name:'Product1'})

MERGE (c:Category {name:'Gadgets'})

CREATE (p)-[:BELONGS_TO {added_date: date('2025-10-14')}]->(c);

// 33

MATCH (u:User {name:'Alice'})

SET u.email = 'alice_new@example.com';

// 34

MATCH (r:Review {name:'Review1'})

SET r.rating = 4, r.date = date('2025-10-15');

// 35

MATCH (c:User {name:'Charlie'}), (p:Product {name:'Product3'})

CREATE (c)-[:BOUGHT {date: date('2025-10-14'), quantity: 1, price_paid: 39.99}]->(p);

// 36

MATCH (p:Product {name:'Product3'})-[m:MADE_BY]->(:Brand)

DELETE m

WITH p

MERGE (b:Brand {name:'BrandD'})

CREATE (p)-[:MADE_BY {launch_year: 2025}]->(b);

// 37

MATCH (p:Product {name:'Product2'})

SET p.name = 'BookMaster 2025';
```

```
// 38

MATCH (u:User {name:'Bob'}) SET u.name = 'Robert';

MATCH (u:User {name:'Charlie'}) SET u.name = 'Charles';

// 39

MATCH (p:Product {name:'Product3'})-[r:BELONGS_TO]->(:Category)

DELETE r

WITH p

MERGE (c:Category {name:'Sports'})

CREATE (p)-[:BELONGS_TO {added_date: date('2025-10-14')}]->(c);

// 40

MATCH (p:Product {name:'Product1'})

SET p.discount = '10%';

// 41

MATCH (p:Product {name:'Product50'})

DETACH DELETE p;

// 42

MATCH (r:Review {name:'Review5', rating:2})

DETACH DELETE r;

// 43

MATCH (:User {name:'Alice'})-[b:BOUGHT]->(:Product {name:'Product2'})

DELETE b;

// 44

MATCH (c:Category {name:'OldCategory'})

DETACH DELETE c;

// 45
```

```
MATCH (u:User {name:'Tina'})

DETACH DELETE u;

// 46

MATCH (:Product {name:'Product1'})-[s:SIMILAR_TO]->(:Product {name:'Product3'})

DELETE s;

// 47

MATCH (b:Brand {name:'BrandJ'})<-[:MADE_BY]-(p:Product)

DETACH DELETE p;

// 48

MATCH (:User)-[v:VIEWED]->(p:Product {name:'Product3'})

WHERE v.date < date('2025-01-01')

DELETE v;

// 49

MATCH (p:Product)

WHERE p.name IN ['Product48','Product49']

  AND NOT ( (:User)-[:BOUGHT]->(p) OR (:Review)-[:REVIEWS]->(p) )

DETACH DELETE p;

// 50

MATCH (r:Review {name:'Review3'})

DETACH DELETE r;
```

Github Link: https://github.com/meghana1653/Data-Engineering/blob/main/lab-7.cypher