CREATE (m1:Member{name: 'Emma', joinDate: '2023-01-15'})-[:BORROWED {date: '2023-05-01'}] ->(b1:Book{title: 'Harry Potter', genre: 'Fantasy', publishYear: 1997})

CREATE (m1)-[:BORROWED {date: '2023-05-01'}]->(b3:Book{title: 'The Hobbit',genre: 'Fantasy',publishYear: 1937})

CREATE (m2:Member {name: 'Jack', joinDate: '2023-02-20'})-[:BORROWED {date: '2023-05-10'}]->(b2:Book {title: 'Lord of the Rings', genre: 'Fantasy', publishYear: 1954})

CREATE (m3:Member {name: 'Sophie', joinDate: '2023-03-10'})-[:BORROWED {date: '2023-07-01'}]->(b1)

CREATE (m4:Member {name: 'David', joinDate: '2023-04-05'})

MATCH (m:Member {name: 'Emma'})-[r:BORROWED]->(b:Book)

RETURN b.title, r.date

MATCH (m:Member)

WHERE NOT (m)-[:BORROWED]->()

RETURN m.name

MATCH (m:Member)-[:BORROWED]->(b:Book)

WITH b, COUNT(\*) as borrowCount

WHERE borrowCount > 1

RETURN b.title, borrowCount

MATCH (m:Member)-[:BORROWED]->(b:Book)

WITH b, COUNT(\*) as borrowCount

WHERE borrowCount > 1

RETURN b.title, borrowCount

MATCH (m:Member)-[:BORROWED]->(b:Book)

WITH b, COUNT(\*) as borrowCount

WHERE borrowCount > 1

RETURN b.title, borrowCount

CREATE (p1:Post { title: 'Vacation', date: '2023-05-15', type: 'photo'})

CREATE (p2:Post { title: 'Party', date: '2023-06-20', type: 'photo'})

CREATE (p3:Post { title: 'Graduation', date: '2023-07-01', type: 'photo'})

MATCH (u1:User { name: 'Alex'}),

      (p1:Post { title: 'Vacation'})

 CREATE (u1)-[:POSTED]->(p1)

MATCH (u1:User { name: 'Alex'}),

      (p2:Post { title: 'Party'})

 CREATE (u1)-[:POSTED]->(p2)

Match(u2:User{name:'Maya'}),

 (p3:Post { title: 'Graduation'})

 CREATE (u2)-[:POSTED]->(p3)

Match(u2:User{name:'Maya'}),

 (p3:Post { title: 'Graduation'})

 CREATE (u2)-[:POSTED]->(p3)

Match(u3:User{name:'Ryan'}),

 (p2:Post { title: 'Party'})

CREATE (u3)-[:COMMENTED {

text: 'Great party!',

date: '2023-06-21'

}]->(p2)

MATCH (u4:User {name: 'Priya'}),

 (p1:Post { title: 'Vacation'}),

  (p2:Post { title: 'Party'})

CREATE (u4)-[:LIKED {date: '2023-05-16'}]->(p1)

CREATE (u4)-[:LIKED {date: '2023-06-22'}]->(p2)

Match (u5:User{name:'Sam'}),

(u1:User { name: 'Alex'}),

(u2:User { name: 'Maya'}),

(u3:User {name: 'Ryan'}),

(u4:User {name: 'Priya'})

CREATE (u5)-[:FOLLOWS {since: '2023-04-10'}]->(u1)

CREATE (u5)-[:FOLLOWS {since: '2023-04-10'}]->(u2)

CREATE (u5)-[:FOLLOWS {since: '2023-04-10'}]->(u3)

CREATE (u5)-[:FOLLOWS {since: '2023-04-10'}]->(u4)

MATCH (u:User {name: 'Alex'})-[:POSTED]->(p:Post)

RETURN p.title, p.date

MATCH (u:User)-[:LIKED]->(p:Post)<-[:POSTED]-(poster:User)

WHERE poster.name = 'Alex'

RETURN DISTINCT u.name

MATCH (u:User)

WHERE NOT (u)-[:POSTED]->()

RETURN u.name

MATCH (follower:User)-[r:FOLLOWS]->(followed:User)

RETURN follower.name, followed.name, r.since

MATCH (u:User)

WHERE (u)-[:POSTED]->() AND (u)-[:COMMENTED]->()

RETURN u.name

MATCH (post:Post)<-[like:LIKED]-()

RETURN post.title, COUNT(like) as likeCount

ORDER BY likeCount DESC

MATCH (u:User)

OPTIONAL MATCH (u)-[:POSTED]->(p:Post)

OPTIONAL MATCH (u)-[:COMMENTED]->(c:Post)

OPTIONAL MATCH (u)-[:LIKED]->(l:Post)

RETURN

u.name,

COUNT(DISTINCT p) as posts,

COUNT(DISTINCT c) as comments,

COUNT(DISTINCT l) as likes,

COUNT(DISTINCT p) + COUNT(DISTINCT c) + COUNT(DISTINCT l) as totalActivity

ORDER BY totalActivity DESC

db.Products.find({Stock:{$gt:10}})

db.Products.updateOne({ProductID: 7}, {$set:{Price:19.99}})

db.Products.deleteOne({ProductID: 2})

db.Products.find({Stock:{$gt:0}}).count()

db.Products.find().sort({Price:-1})

db.Products.find({Price:{$in:[10,15]}})

db.Products.find({Category: "Home Appliances", Stock:{$gt:30}})

db.Products.updateOne({ProductID: 4}, {$set:{Category: "Furniture"}})

db.Products.find({Stock:{$nin:[10,15]}})

db.Products.find({Stock:{$lt:5}, Price:{$gt:10}})

db.Products.find({Name:{$regex:/c/}})

db.Products.find().skip(2).limit(5)

db.Customers.updateOne({CustomerID: 4}, {$set:{AccountBalance: 5000}})

db.Customers.deleteOne({CustomerID: 2})

db.Customers.find({AccountBalance:{$gt: 1000}}).count()

db.Customers.find().sort({AccountBalance:1})

db.Customers.find({AccountType:{$in:["Checking", "Business"]}})

db.Customers.find({AccountBalance:{$gt:2000, $lt:5000}})

db.Customers.updateMany({AccountType: "Savings"}, {$inc:{AccountBalance: 1000}})

db.Customers.find({AccountBalance:{$gt:5000}})

db.Customers.find({AccountType: "Checking", AccountBalance:{$lt:1000}})

db.Customers.find({AccountType:{$in:["Checking", "Savings"]}})

db.Customers.find({AccountBalance:{$gt:3000, $lt:7000}})

db.Customers.updateMany({AccountBalance:{$lt:1000}}, {$set:{AccountBalance: 1500}})

db.Customers.deleteMany({})

const author = db.Authors.findOne({Name: "J.K. Rowling"}) db.Books.find({AuthorID: author.AuthorID})

db.Books.updateOne({BookID: 3}, {$set:{PublishedYear: 2022}})

db.Books.deleteOne({BookID: 2})

db.Books.find({Genre: "Science Fiction"}).count()

> db.Books.find({PublishedYear:{$gt:2000, $lt:2010}})

db.Books.find({Genre:{$in:["Thriller", "Romance", "Adventure"]}})

const author\_name = db.Authors.findOne({DateOfBirth:{$gt:1980}, Name: "Ruskin Bond"}) db.Books.find({AuthorID:author\_name.AuthorID})

db.Books.find().sort({Title: 1})

db.Books.aggregate([{$group:{\_id:"$Genre", avgPublicationYear:{$avg: "$PublishedYear"}}}])

const book = db.Books.findOne({Genre: "Horror"}) db.Authors.find({AuthorID: book.AuthorID})

CREATE VIEW AVAILABLE\_BOOKS AS SELECT copies FROM books WHERE copies > 5;

SELECT b.borrower\_id, b.name, b.address, b.phone, b.membership\_type FROM borrowers b JOIN issues u ON b.borrower\_id = u.borrower\_id GROUP BY u.borrower\_id HAVING COUNT(u.borrower\_id) > 2;

SELECT b.name FROM borrowers b JOIN issues u ON b.borrower\_id = u.borrower\_id WHERE u.return\_date IS NULL;

MOVIE>

ALTER TABLE movies MODIFY COLUMN genre VARCHAR(45) NOT NULL;

ALTER TABLE customer ADD CONSTRAINT unique\_email UNIQUE customer(email);

ALTER TABLE rentals ADD CONSTRAINT movie\_id FOREIGN KEY (movie\_id) REFERENCES movies(movie\_id);

CREATE INDEX rating\_index ON movies(rating);

SELECT ROUND(AVG(rating),2) AS avg\_rating FROM movies;

UPDATE movies SET genre = "Uncategorized" WHERE rating < 5;

DELETE FROM customer WHERE membership\_type IS NULL;

SELECT c.name FROM customer c JOIN rentals r ON c.customer\_id = r.customer\_id WHERE r.return\_date IS NULL;

SELECT m.title AS movie, COUNT(r.movie\_id) AS no\_of\_times\_rented FROM movies m JOIN rentals r ON m.movie\_id = r.movie\_id GROUP BY r.movie\_id;

CREATE INDEX type\_index ON transactions(type);

UPDATE customers SET balance = balance + ( SELECT SUM(t.amount) FROM transactions t WHERE t.account\_no = customers.account\_no AND t.type = "Deposit" AND t.amount > 100000

SELECT c.name, SUM(t.amount) AS total\_transaction\_amount FROM customers c JOIN transactions t ON c.account\_no = t.account\_no GROUP BY c.name;

SELECT LEFT(name, 3) AS first\_three\_characters FROM customers;

START TRANSACTION; SELECT \* FROM customers WHERE name = "Jane Doe"; SAVEPOINT before\_update; UPDATE customers SET balance = 50000 WHERE name = "Jane Doe"; SAVEPOINT after\_update; SELECT \* FROM customers WHERE name = "Jane Doe";

SELECT ROUND(AVG(amount), 2) AS avg\_transaction\_amt FROM transactions WHERE transaction\_date BETWEEN "2023-01-01" AND "2023-12-30" AND type = "Withdraw";

ALTER TABLE customers MODIFY COLUMN phone VARCHAR(15) NOT NULL;

SELECT c.customer\_id, c.name, c.account\_no, c.phone, c.balance FROM customers c JOIN transactions t ON c.account\_no = t.account\_no WHERE t.amount > 100000;

SELECT c.name, SUM(t.amount) AS total\_transaction\_amounts FROM customers c JOIN transactions t ON c.account\_no = t.account\_no GROUP BY c.name;

DELETE FROM customers WHERE account\_no NOT IN ( SELECT DISTINCT account\_no FROM transactions WHERE transaction\_date >= DATE\_SUB(CURDATE(), INTERVAL 1 YEAR)