Islamic University of Technology

Lab-09 Report: Neo4j Scenario

Submitted by:

Name : Mashrur Ahsan

ID : 200042115

Program : SWE

Department : CSE

Course Code : CSE 4410

The Scenario:

- Let's assume you want to build a recommendation engine for your online bookshop. Your online bookshop sells a variety of books and you want to improve the customer's buying experience by recommending books that they are likely to purchase by analyzing shared personal details and monitoring which books go together in one's purchase list.

  This type of scenario generally includes four types of nodes-

  - Customer: Contains information about customers such as customer ID, name, phone_no, and demographic information like age, gender, country etc.
  - Genre: It helps to filter different books according to the genre.
  - Author: Contains information about authors such as name, country, date_of_birth etc.
  - Book: Contains information like title, published_year, language, page_count, price etc.

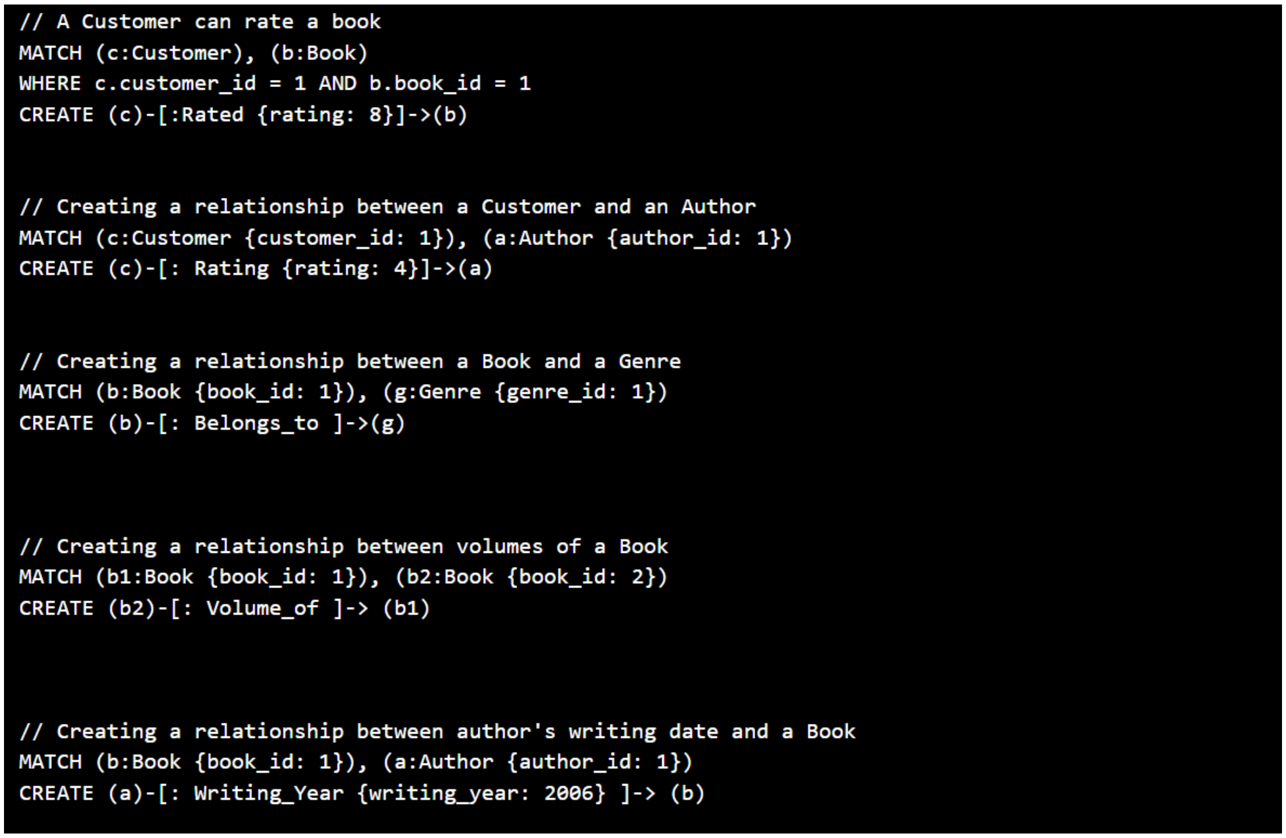
  And the relations are the following-

  - Customers purchase or rate books. The purchase information also includes purchasing_date and amount.
  - Customer can also rate authors(this is different from rating a book).
  - Books can be of different genres.
  - Books can have multiple volumes.
  - Authors write books. And this includes the writing_year.

Tasks:

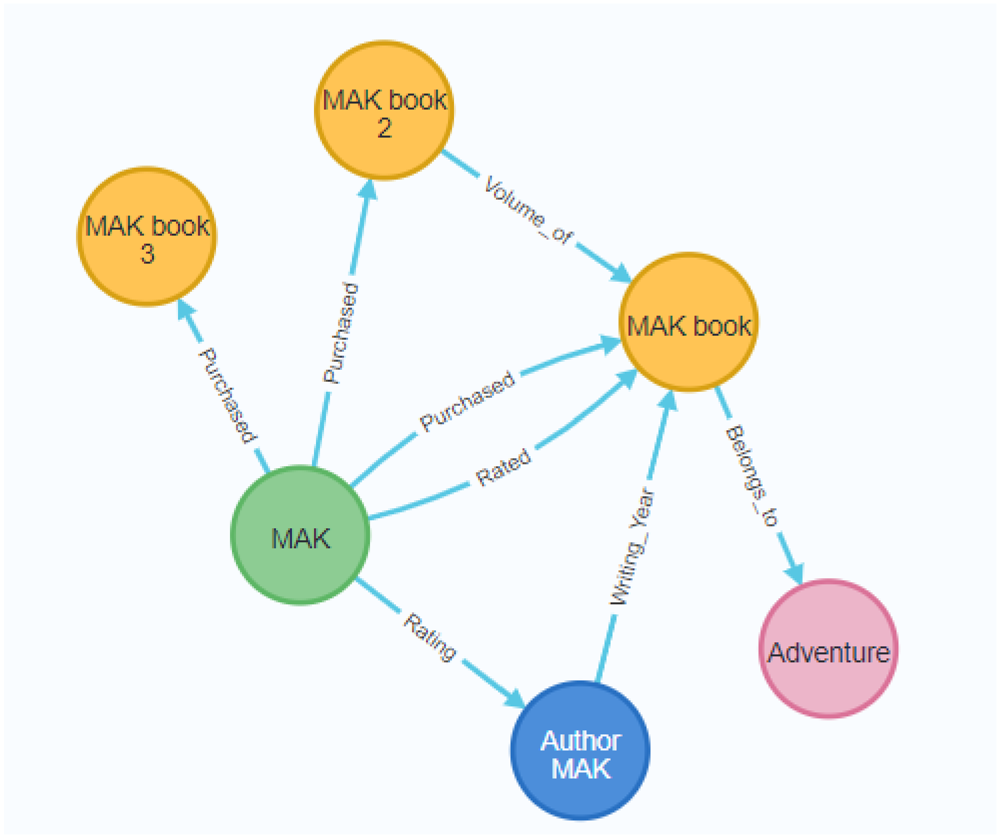
1. Create necessary nodes and relations with properties.

```
// Creating a Customer node
CREATE (:Customer {customer_id: 1, name: 'MAK', phone_no: '12345678', age: 21, gender: 'male', country: 'BD'})

// Creating a Genre node
CREATE (:Genre {genre_id: 1, name: 'Fiction'})

// Creating an Author node
CREATE (:Author {author_id: 1, name: 'Author MAK', country: 'UK', date_of_birth: '1995-07-31'})

// Creating a Book node
CREATE (:Book {book_id: 1, title: 'MAK book', published_year: 2007, language: 'English', page_count: 223, price: 29.55})
CREATE (:Book {book_id: 2, title: 'MAK book 2', published_year: 2009, language: 'English', page_count: 653, price: 99.55})
CREATE (:Book {book_id: 3, title: 'MAK book 3', published_year: 2011, language: 'English', page_count: 573, price: 299.99})


// Creating a relationship between a Customer and a Book
MATCH (c:Customer {customer_id: 1}), (b:Book {book_id: 1})
CREATE (c)-[:Purchased {purchasing_date: '2002-03-15', amount: 10.99}]->(b)

MATCH (c:Customer {customer_id: 1}), (b:Book {book_id: 2})
CREATE (c)-[:Purchased {purchasing_date: '2002-01-01', amount: 29.55}]->(b)

MATCH (c:Customer {customer_id: 1}), (b:Book {book_id: 3})
CREATE (c)-[:Purchased {purchasing_date: '2022-03-15', amount: 299.99}]->(b)
```



```
// A Customer can rate a book
MATCH (c:Customer), (b:Book)
WHERE c.customer_id = 1 AND b.book_id = 1
CREATE (c)-[:Rated {rating: 8}]->(b)


// Creating a relationship between a Customer and an Author
MATCH (c:Customer {customer_id: 1}), (a:Author {author_id: 1})
CREATE (c)-[: Rating {rating: 4}]->(a)


// Creating a relationship between a Book and a Genre
MATCH (b:Book {book_id: 1}), (g:Genre {genre_id: 1})
CREATE (b)-[: Belongs_to ]->(g)



// Creating a relationship between volumes of a Book
MATCH (b1:Book {book_id: 1}), (b2:Book {book_id: 2})
CREATE (b2)-[: Volume_of ]-> (b1)



// Creating a relationship between author's writing date and a Book
MATCH (b:Book {book_id: 1}), (a:Author {author_id: 1})
CREATE (a)-[: Writing_Year {writing_year: 2006} ]-> (b)
```

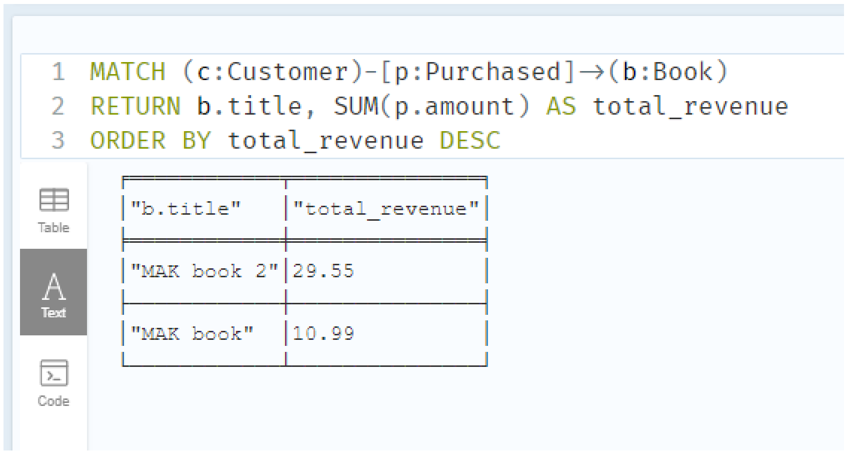
<u>More relationships could've been made. But this was enough to show that all of the queries worked properly.</u>



2. Cypher Query

(a) Find the total revenue generated by each book.

```
// a
MATCH (c:Customer)-[p:Purchased]->(b:Book)
RETURN b.title, SUM(p.amount) AS total_revenue
ORDER BY total_revenue DESC
```

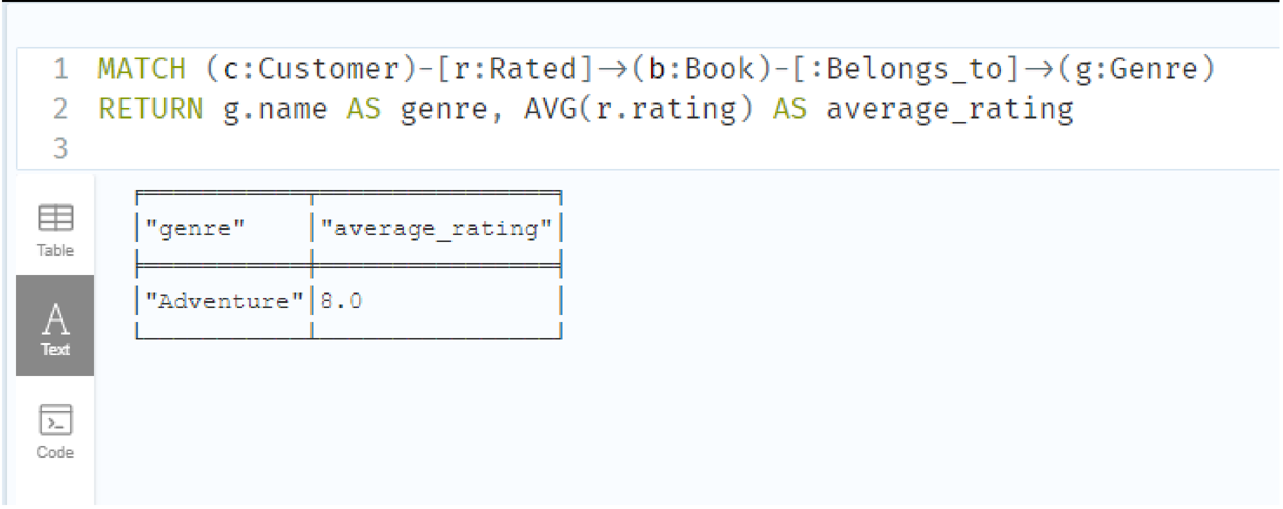


```
MATCH (c:Customer)-[p:Purchased]→(b:Book)
RETURN b.title, SUM(p.amount) AS total_revenue
ORDER BY total_revenue DESC
```

```
╒════════════╤═══════════════╕
│"b.title"   │"total_revenue"│
╞════════════╪═══════════════╡
│"MAK book 2"│29.55          │
├────────────┼───────────────┤
│"MAK book"  │10.99          │
└────────────┴───────────────┘
```

(b) Find the average rating for each genre.

```
// b
MATCH (c:Customer)-[r:Rated]->(b:Book)-[:Belongs_to]->(g:Genre)
RETURN g.name AS genre, AVG(r.rating) AS average_rating
```

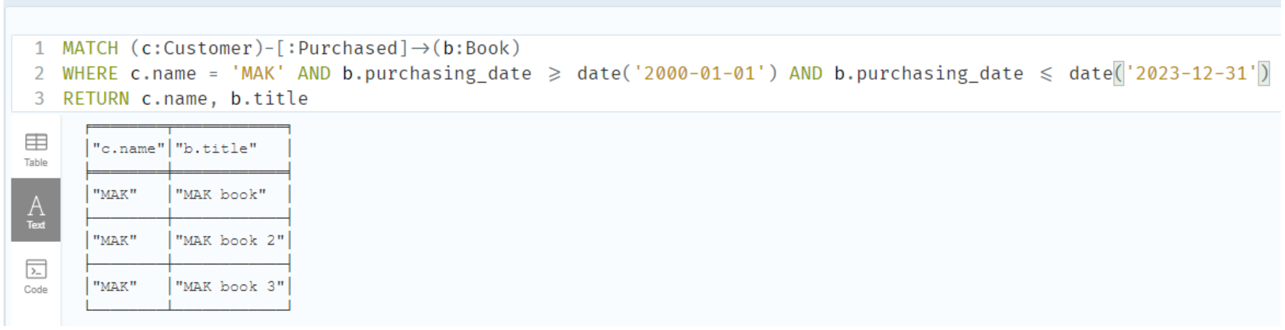


```
MATCH (c:Customer)-[r:Rated]→(b:Book)-[:Belongs_to]→(g:Genre)
RETURN g.name AS genre, AVG(r.rating) AS average_rating

```

```
╒═══════════╤════════════════╕
│"genre"    │"average_rating"│
╞═══════════╪════════════════╡
│"Adventure"│8.0             │
└───────────┴────────────────┘
```

(c) Find books purchased by a customer 'N' within a specific time range.

```
// c
MATCH (c:Customer)-[:Purchased]->(b:Book)
WHERE c.name = 'MAK' AND b.purchasing_date >= date('2000-01-01') AND b.purchasing_date <= date('2023-12-31')
RETURN c.name, b.title

MATCH (c:Customer {customer_id: 1})-[:Purchased]->(b:Book)
WHERE b.purchasing_date >= date('2002-01-01') AND b.purchasing_date <= date('2022-03-15')
RETURN c.name, b.title
```

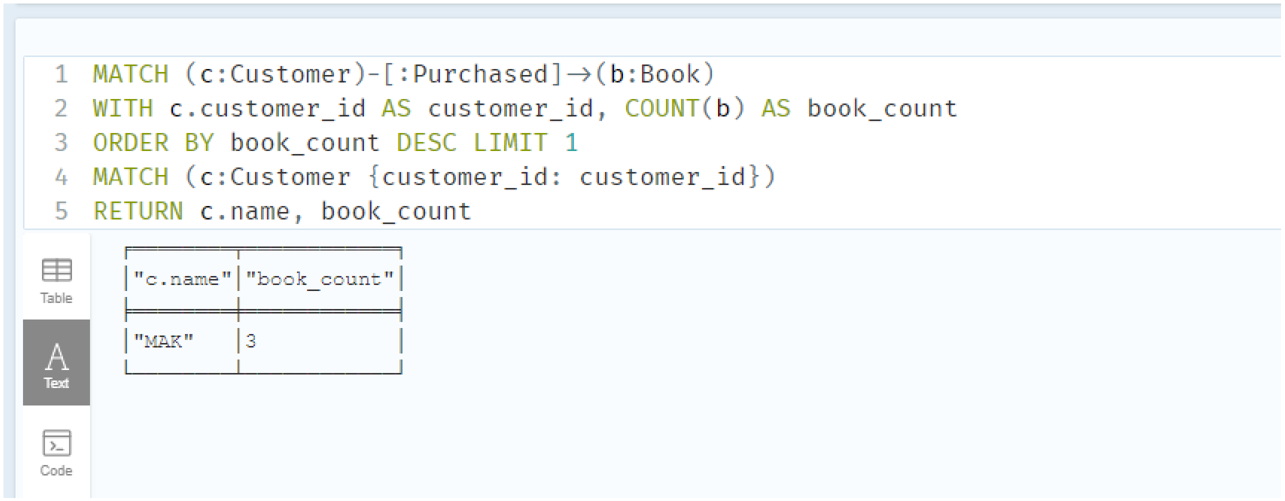


```
MATCH (c:Customer)-[:Purchased]→(b:Book)
WHERE c.name = 'MAK' AND b.purchasing_date ≥ date('2000-01-01') AND b.purchasing_date ≤ date('2023-12-31')
RETURN c.name, b.title
```

```
╒════════╤═════════════╕
│"c.name"│"b.title"    │
╞════════╪═════════════╡
│"MAK"   │"MAK book"   │
├────────┼─────────────┤
│"MAK"   │"MAK book 2" │
├────────┼─────────────┤
│"MAK"   │"MAK book 3" │
└────────┴─────────────┘
```

(d) Find the customer who buys the maximum number of books.

```
// d
MATCH (c:Customer)-[:Purchased]->(b:Book)
WITH c, COUNT(b) AS num_books
ORDER BY num_books DESC
LIMIT 1
RETURN c.name, num_books

MATCH (c:Customer)-[:Purchased]->(b:Book)
WITH c.customer_id AS customer_id, COUNT(b) AS num_books
ORDER BY num_books DESC
LIMIT 1
MATCH (c:Customer {customer_id: customer_id})
RETURN c.name, num_books
```

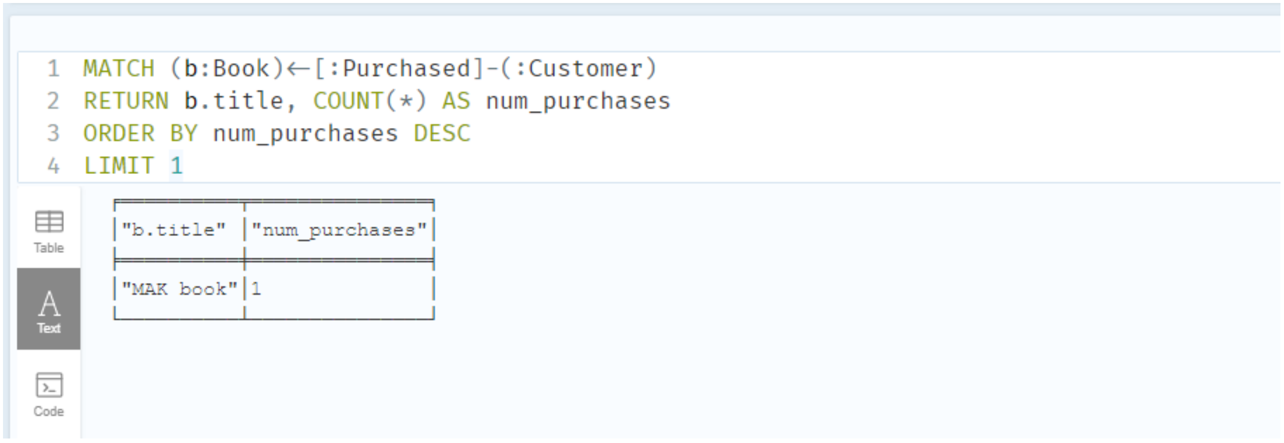


```
MATCH (c:Customer)-[:Purchased]→(b:Book)
WITH c.customer_id AS customer_id, COUNT(b) AS book_count
ORDER BY book_count DESC LIMIT 1
MATCH (c:Customer {customer_id: customer_id})
RETURN c.name, book_count
```

```
╒════════╤════════════╕
│"c.name"│"book_count"│
╞════════╪════════════╡
│"MAK"   │3           │
└────────┴────────────┘
```

(e) Find the best-seller books by the number of purchases.

```
// e
MATCH (b:Book)<-[:Purchased]-()
RETURN b.title, COUNT(*) AS num_purchases
ORDER BY num_purchases DESC
LIMIT 1
```

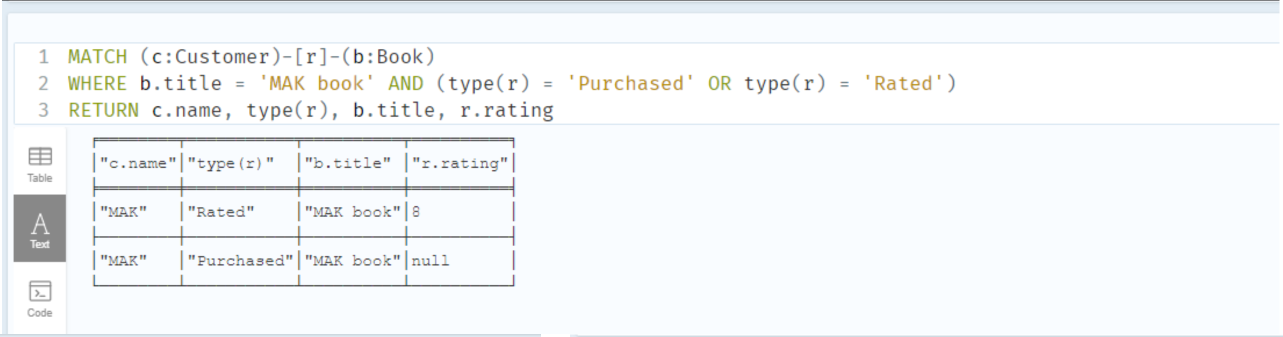


```
MATCH (b:Book)←[:Purchased]-(:Customer)
RETURN b.title, COUNT(*) AS num_purchases
ORDER BY num_purchases DESC
LIMIT 1
```

```
│"b.title"  │"num_purchases"│
│"MAK book" │1              │
```

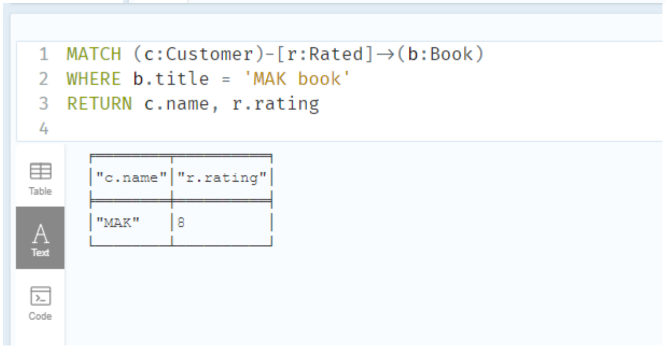
(f) Find the customer who bought or rated a certain book. for example 'A'

```
// f
MATCH (c:Customer)-[r:Rated]->(b:Book)    // just rated
WHERE b.title = 'MAK book'
RETURN c.name, r.rating

MATCH (c:Customer)-[:Purchased]->(b:Book {title: 'MAK book'})   // just bought
RETURN c.name, b.title

MATCH (c:Customer)-[r]-(b:Book)
WHERE b.title = 'MAK book' AND (type(r) = 'Purchased' OR type(r) = 'Rated')   // for both
RETURN c.name, type(r), b.title, r.rating
```



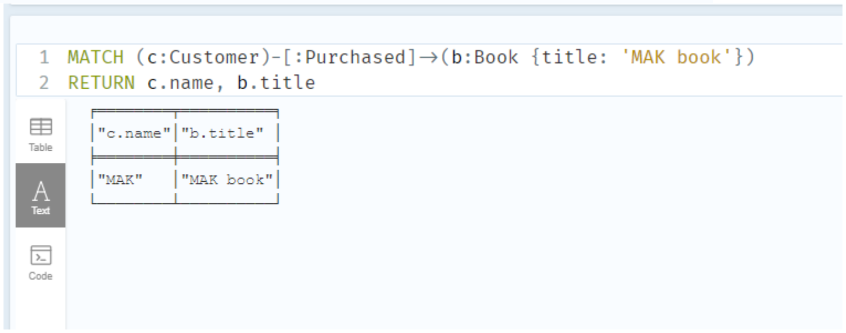
```
MATCH (c:Customer)-[r]-(b:Book)
WHERE b.title = 'MAK book' AND (type(r) = 'Purchased' OR type(r) = 'Rated')
RETURN c.name, type(r), b.title, r.rating
```

```
│"c.name"│"type(r)"  │"b.title"  │"r.rating"│
│"MAK"   │"Rated"    │"MAK book" │8         │
│"MAK"   │"Purchased"│"MAK book" │null      │
```



```
MATCH (c:Customer)-[r:Rated]→(b:Book)
WHERE b.title = 'MAK book'
RETURN c.name, r.rating
```

```
│"c.name"│"r.rating"│
│"MAK"   │8         │
```

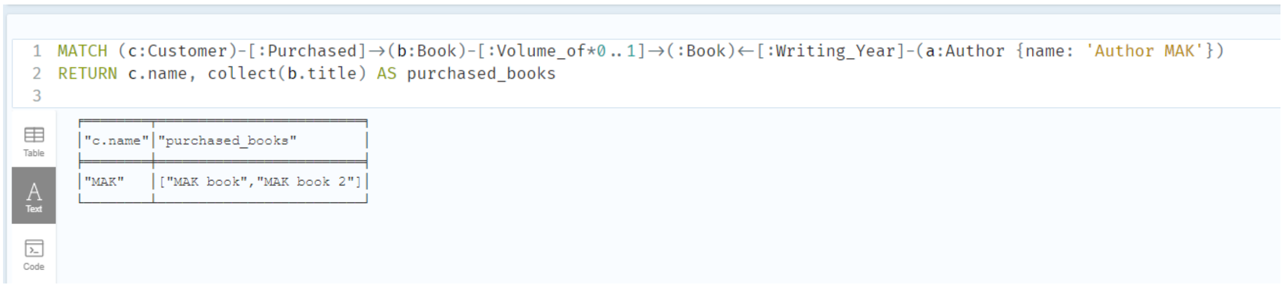


```
MATCH (c:Customer)-[:Purchased]→(b:Book {title: 'MAK book'})
RETURN c.name, b.title
```

```
│"c.name"│"b.title"  │
│"MAK"   │"MAK book" │
```

(g) Find the customer who bought the books of a certain author. for example 'X'

```
// g
MATCH (c:Customer)-[:Purchased]->(b:Book)-[:Volume_of*0..1]->(:Book)<-[:Writing_Year]-(a:Author {name: 'Author MAK'})
RETURN c.name, collect(b.title) AS purchased_books
```



```
MATCH (c:Customer)-[:Purchased]→(b:Book)-[:Volume_of*0..1]→(:Book)←[:Writing_Year]-(a:Author {name: 'Author MAK'})
RETURN c.name, collect(b.title) AS purchased_books
```

```
╒════════╤═════════════════════╕
│"c.name"│"purchased_books"    │
╞════════╪═════════════════════╡
│"MAK"   │["MAK book","MAK book 2"]│
└────────┴─────────────────────┘
```

(h) Find books frequently purchased together.

```
// h
MATCH (b1:Book)-[:Purchased]->(c:Customer)-[:Purchased]->(b2:Book)
WHERE b1.book_id < b2.book_id
WITH collect(distinct b1.book_id) as b1_ids, collect(distinct b2.book_id) as b2_ids, count(distinct c.customer_id) as freq
WHERE freq > 1
RETURN b1_ids, b2_ids, freq
ORDER BY freq DESC

MATCH (b1:Book)-[:Purchased]->(c:Customer)<-[:Purchased]-(b2:Book)
WHERE b1 <> b2
WITH b1, b2, count(DISTINCT c) AS freq
WHERE freq > 1
RETURN b1.title, b2.title, freq
ORDER BY freq DESC;
```