Дмитренко ФБ-42мп

Змоделювати наступну предметну область:

- Є: Items, Customers, Orders
- Customer може додати Item(s) до Order (тобто купити Товар)
- У Customer може бути багато Orders
- Item може входити в багато Orders, і у Item є вартість
- Customer може переглядати (view), але при цьому не купувати Items

CREATE (:Customer {id: 1, name: "Alice"});

CREATE (:Customer {id: 2, name: "Bob"});

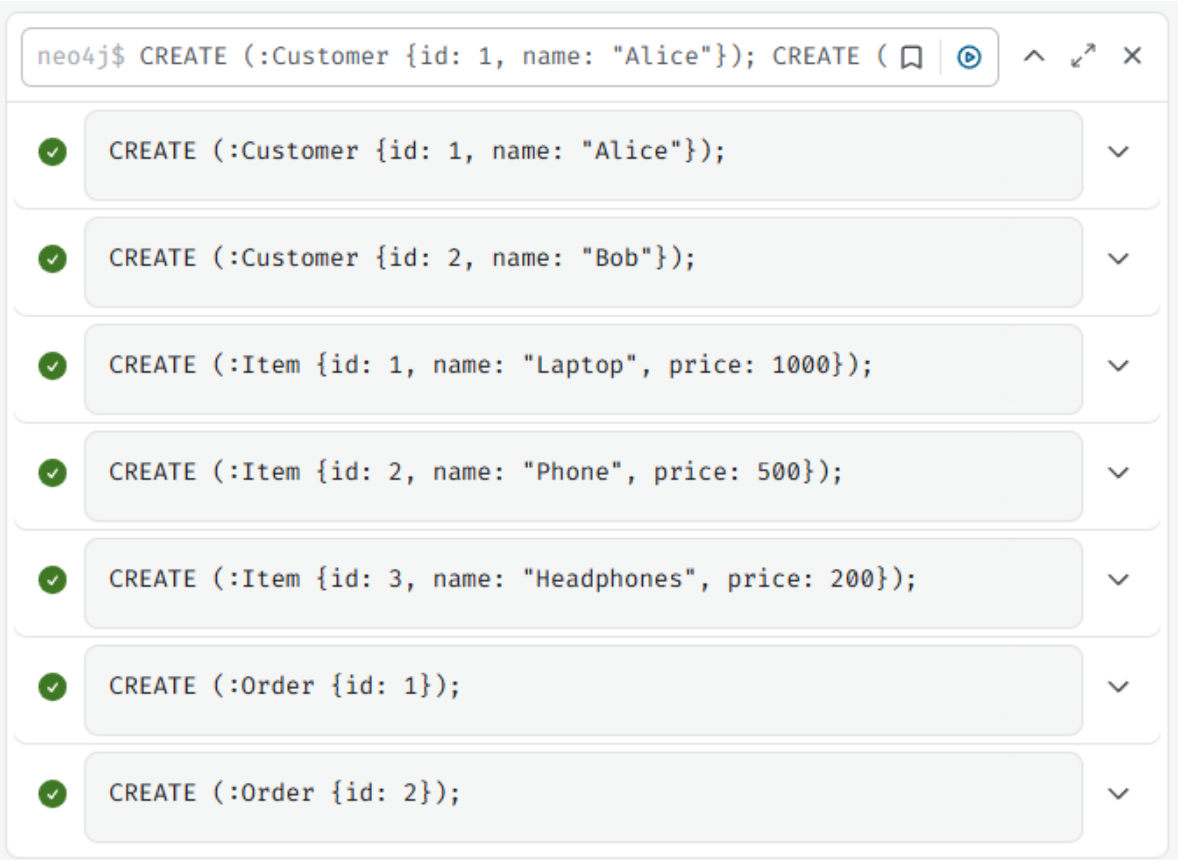
CREATE (:Item {id: 1, name: "Laptop", price: 1000});

CREATE (:Item {id: 2, name: "Phone", price: 500});

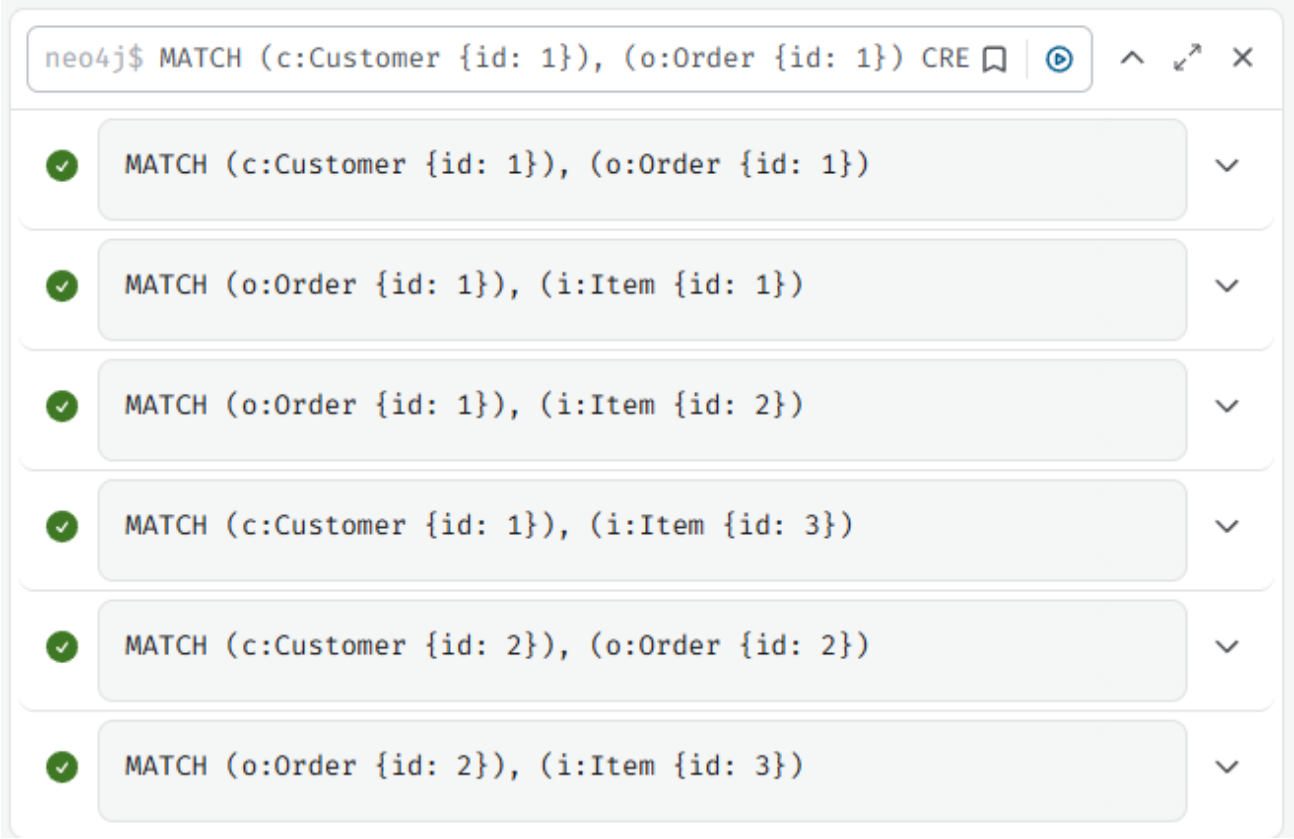
CREATE (:Item {id: 3, name: "Headphones", price: 200});

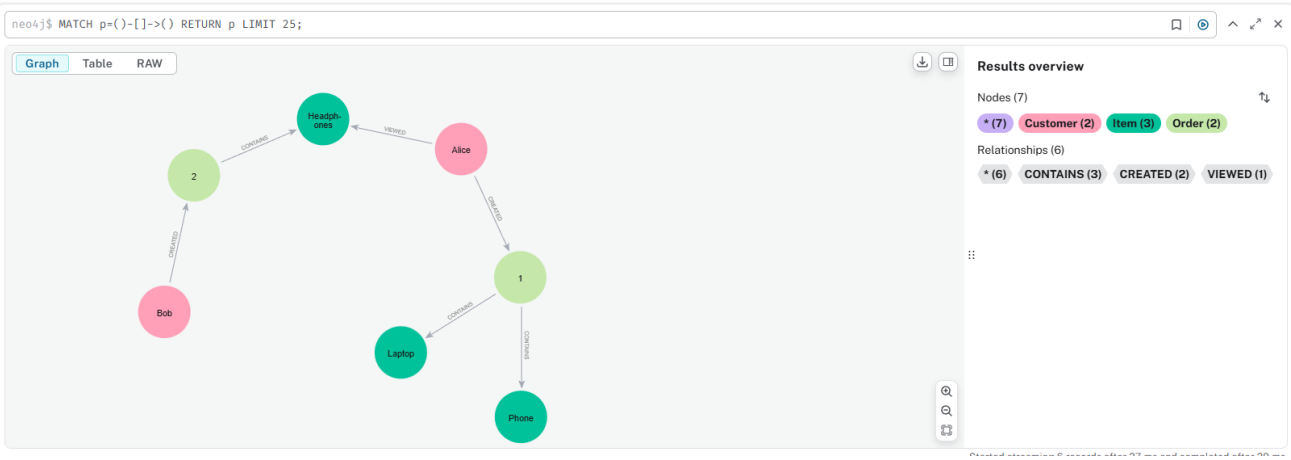
CREATE (:Order {id: 1});

CREATE (:Order {id: 2});



```
MATCH (c:Customer {id: 1}), (o:Order {id: 1})
CREATE (c)-[:CREATED]->(o);
MATCH (o:Order {id: 1}), (i:Item {id: 1})
CREATE (o)-[:CONTAINS]->(i);
MATCH (o:Order {id: 1}), (i:Item {id: 2})
CREATE (o)-[:CONTAINS]->(i);
MATCH (c:Customer {id: 1}), (i:Item {id: 3})
CREATE (c)-[:VIEWED]->(i);
MATCH (c:Customer {id: 2}), (o:Order {id: 2})
CREATE (c)-[:CREATED]->(o);
MATCH (o:Order {id: 2}), (i:Item {id: 3})
CREATE (o)-[:CONTAINS]->(i);
```





Написати наступні види запитів:

- Знайти Items які входять в конкретний Order

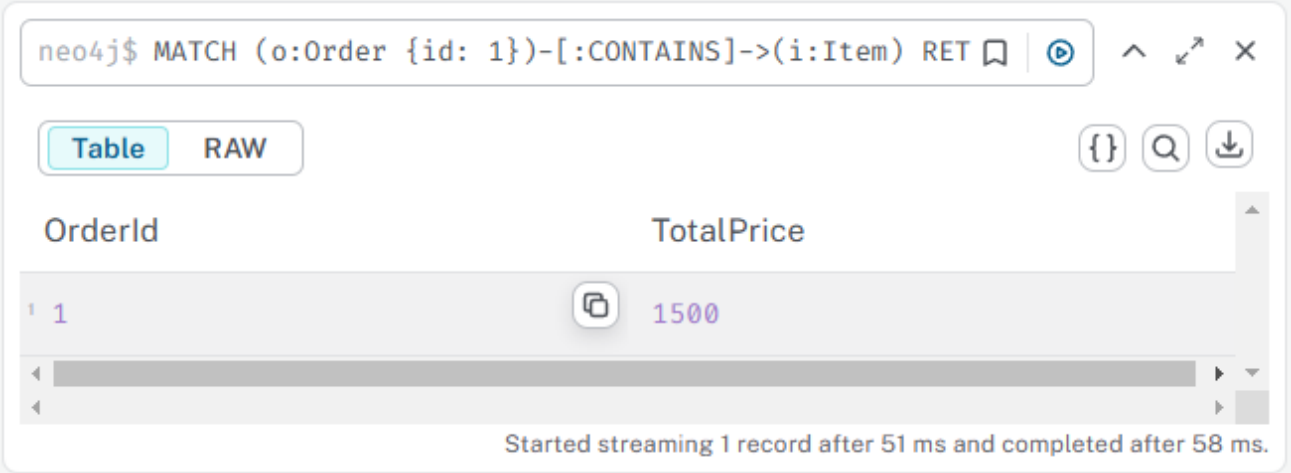
MATCH (o:Order {id: 1})-[:CONTAINS]->(i:Item)

RETURN i;

- Підрахувати вартість конкретного Order

MATCH (o:Order {id: 1})-[:CONTAINS]->(i:Item)

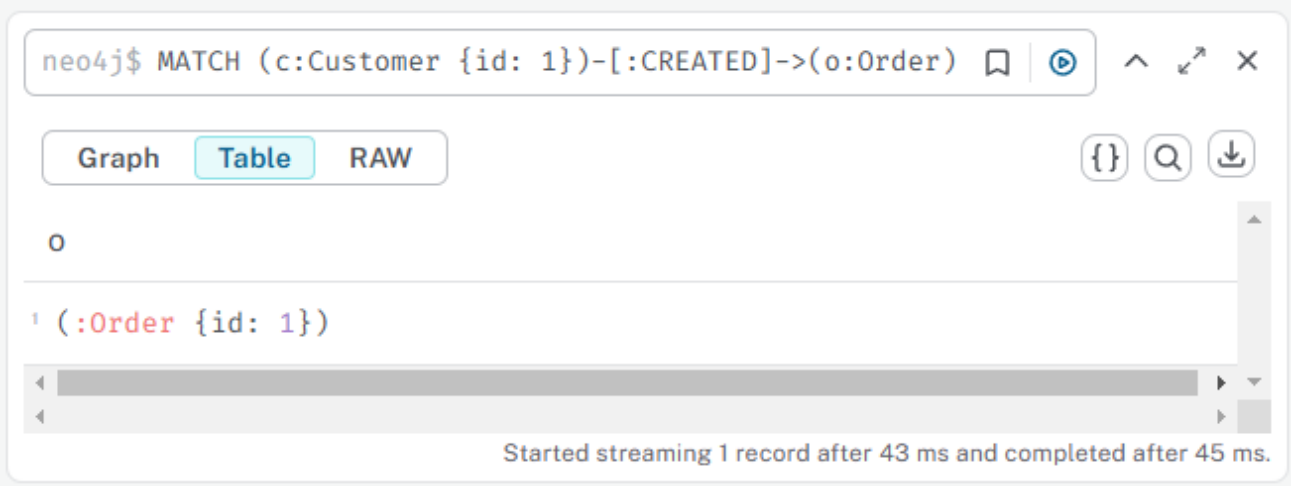
RETURN o.id AS OrderId, SUM(i.price) AS TotalPrice;



- Знайти всі Orders конкретного Customer

MATCH (c:Customer {id: 1})-[:CREATED]->(o:Order)

RETURN o;



- Знайти всі Items куплені конкретним Customer (через Order)

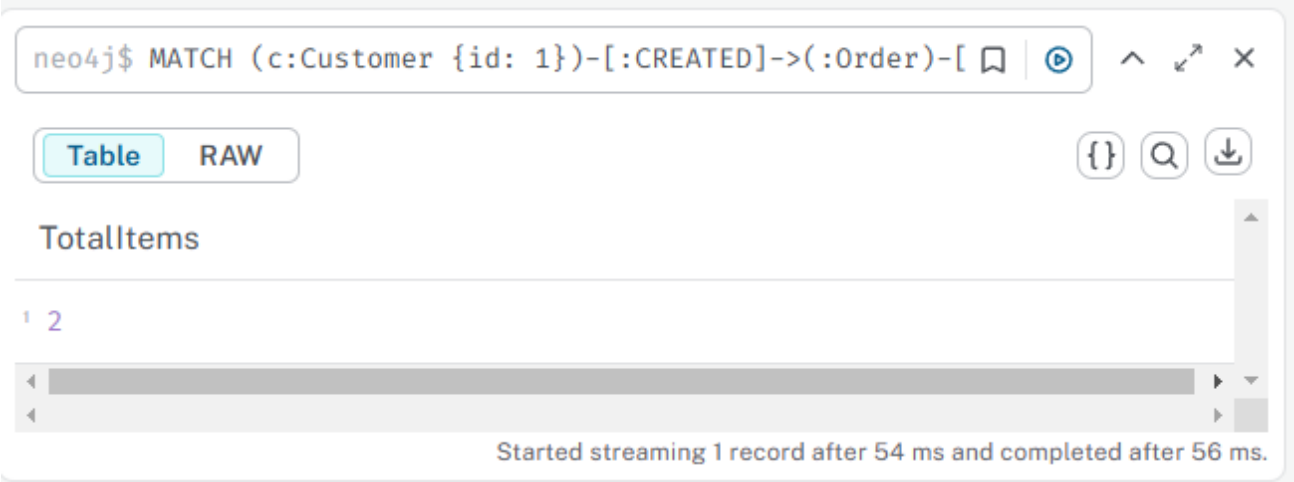
MATCH (c:Customer {id: 1})-[:CREATED]->(:Order)-[:CONTAINS]->(i:Item)

RETURN i;

- Знайти кількість Items куплені конкретним Customer (через Order)

MATCH (c:Customer {id: 1})-[:CREATED]->(:Order)-[:CONTAINS]->(i:Item)

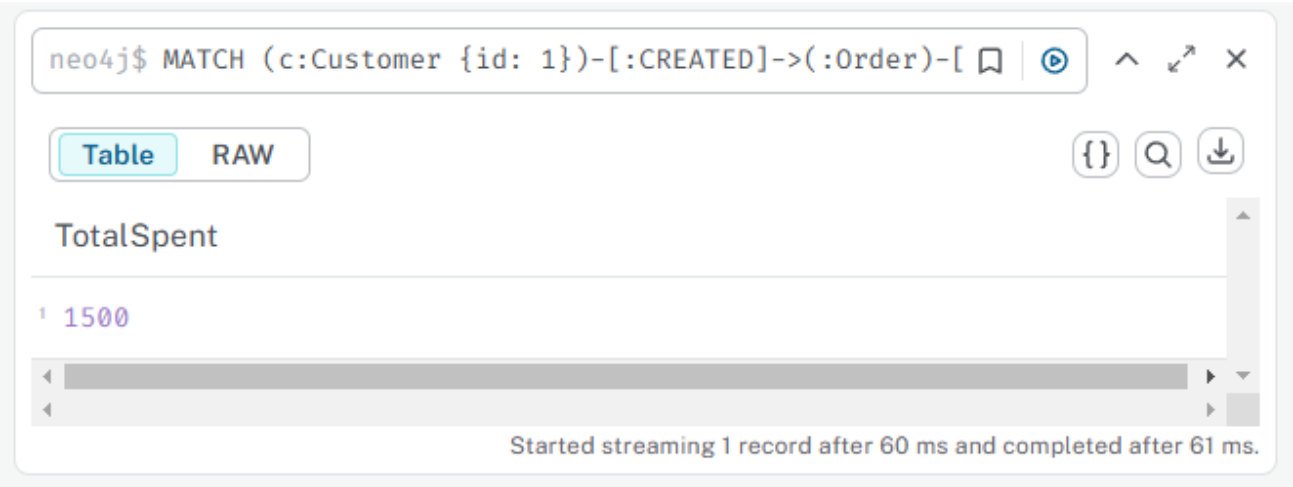
RETURN COUNT(i) AS TotalItems;



- Знайти для Customer на яку суму він придбав товарів (через Order)

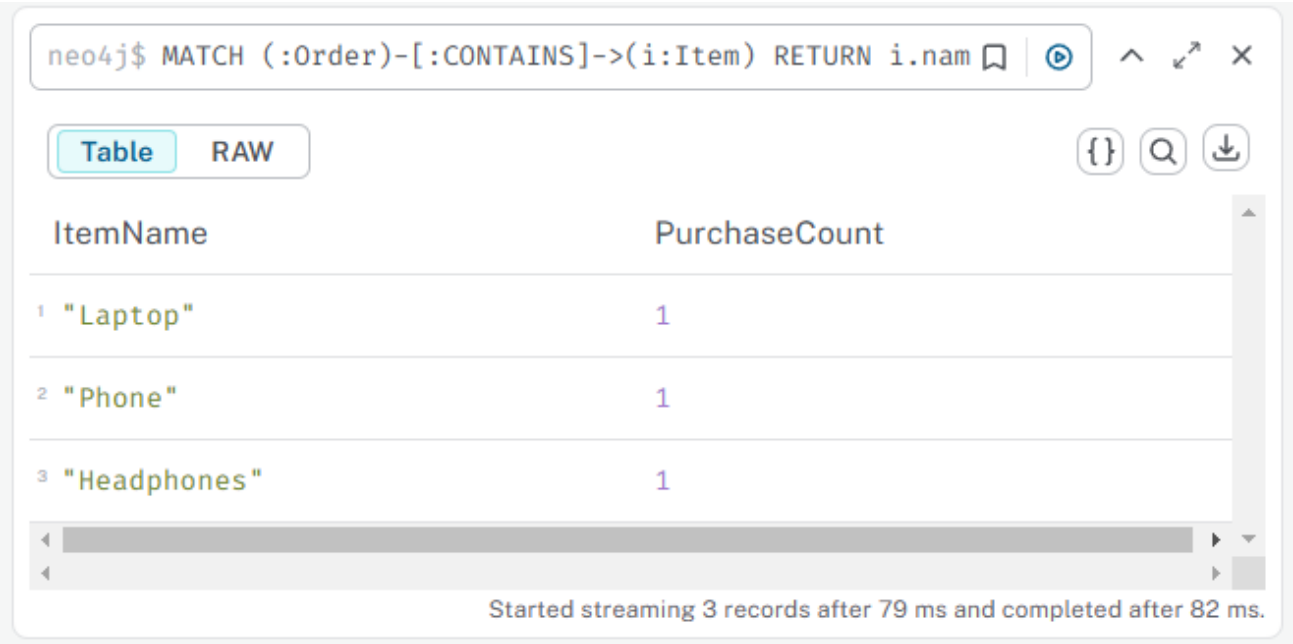
MATCH (c:Customer {id: 1})-[:CREATED]->(:Order)-[:CONTAINS]->(i:Item)

RETURN SUM(i.price) AS TotalSpent;



- Знайті скільки разів кожен товар був придбаний, відсортувати за цим значенням

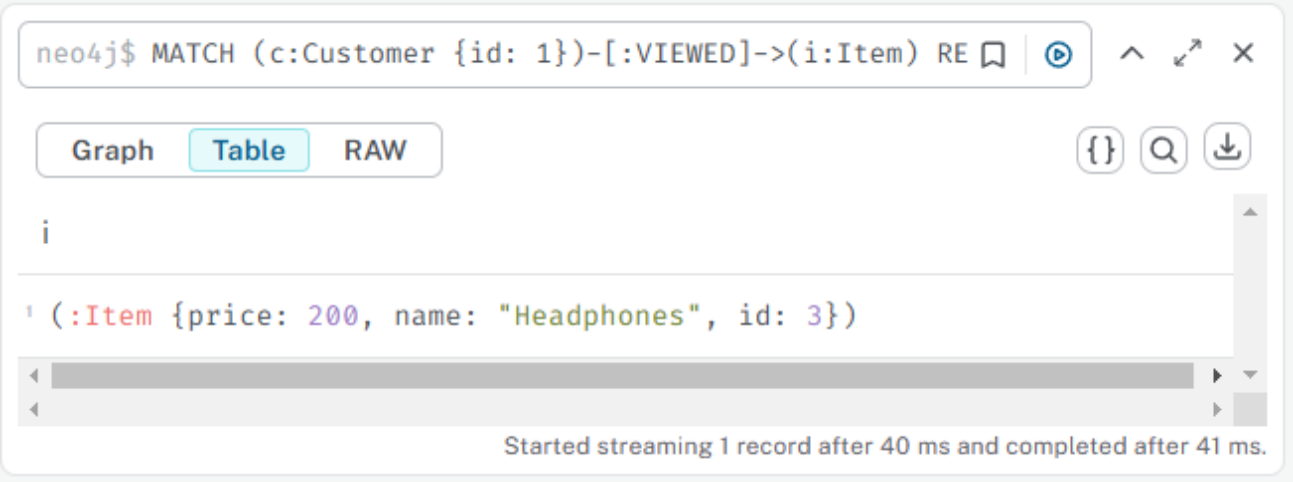
MATCH (:Order)-[:CONTAINS]->(i:Item) RETURN i.name AS ItemName, COUNT(*) AS PurchaseCount ORDER BY PurchaseCount DESC;



- Знайти всі Items переглянуті (view) конкретним Customer

MATCH (c:Customer {id: 1})-[:VIEWED]->(i:Item)

RETURN i;



- Знайти інші Items що купувались разом з конкретним Item (тобто всі Items що входять до Order-s разом з даними Item)

MATCH (:Item {id: 1})<-[:CONTAINS]-(o:Order)-[:CONTAINS]->(i:Item)

WHERE i.id <> 1

RETURN i;

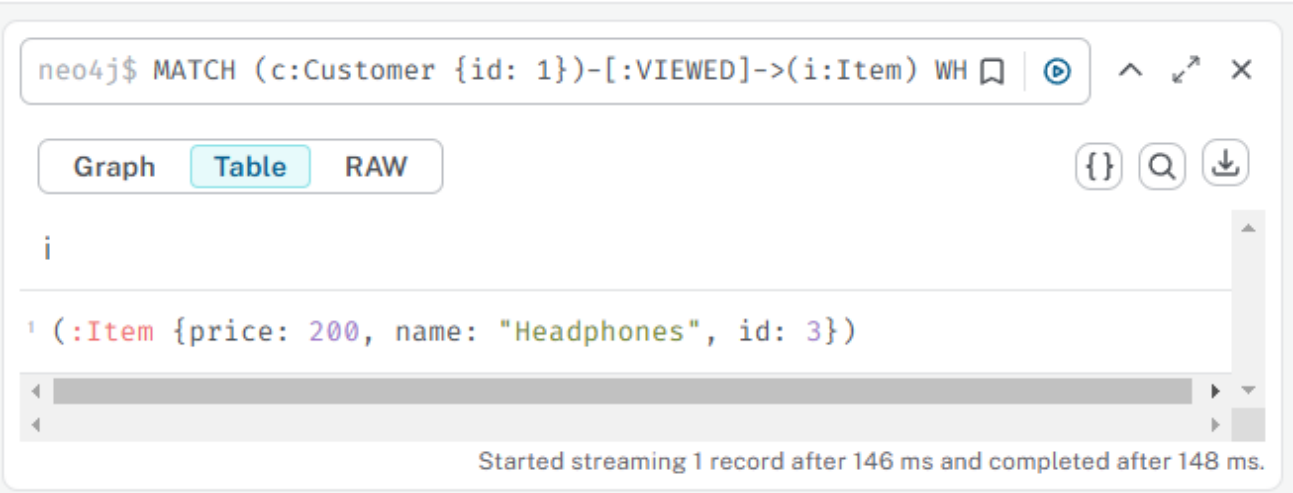
- Знайти Customers які купили даний конкретний Item

MATCH (c:Customer)-[:CREATED]->(:Order)-[:CONTAINS]->(i:Item {id: 1})

RETURN c;

- Знайти для певного Customer(a) товари, які він переглядав, але не купив

```
MATCH (c:Customer {id: 1})-[:VIEWED]->(i:Item)
WHERE NOT EXISTS {
  MATCH (c)-[:CREATED]->(:Order)-[:CONTAINS]->(i)
}
RETURN i;
```



1. Як і в попередніх завданнях, для якогось одного обраного Item додайте поле з кількістю його лайків.

- З 10 окремих клієнтів одночасно запустити інкерементацію каунтеру лайків по 10_000 на кожного клієнта
- зробіть так щоб не було втрат та перевірте щоб фінальне значення було 100_000
- заміряйте час роботи

```
neo4j$ MATCH (i:Item {id: 1}) SET i.likes = 0;
```

Set 1 property

```
neo4j$ MATCH (i:Item {id: 1}) RETURN i.likes;
```

Table RAW

i.likes

1 0

```
┌──(kali㉿kali)-[~/Desktop/pvs]
└─$ /bin/python3 /home/kali/Desktop/pvs/lab3.py
Total likes: 0
Total likes: 100000
time: 2.6375820636749268 seconds
```