Task 2 - Робота з базовими функціями граф-орієнтованої БД на прикладі Neo4j

Завдання:

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
- Customer може переглядати (view), але при цьому не купувати Items

Написати наступні види запитів:

- Знайти Items які входять в конкретний Order
- Знайти всі Orders конкретного Customer
- Знайти всі Items куплені конкретним Customer
- Знайти кількість Items куплені конкретним Customer з усіх Orders
- Знайті скільки разів кожен товар був придбаний
- Знайти всі Items переглянуті (view) конкретним Customer
- Знайти Items що куплені разом з конкретним Item (тобто все Items що входять до Order разом з даними Item)
- Знайти Customers які купили даний конкретний Item
- Знайти всі Items куплені Customer(s) який переглядав даний конкретний Item
- Знайти для певного Customer(a) товари, які він переглядав, але не купив

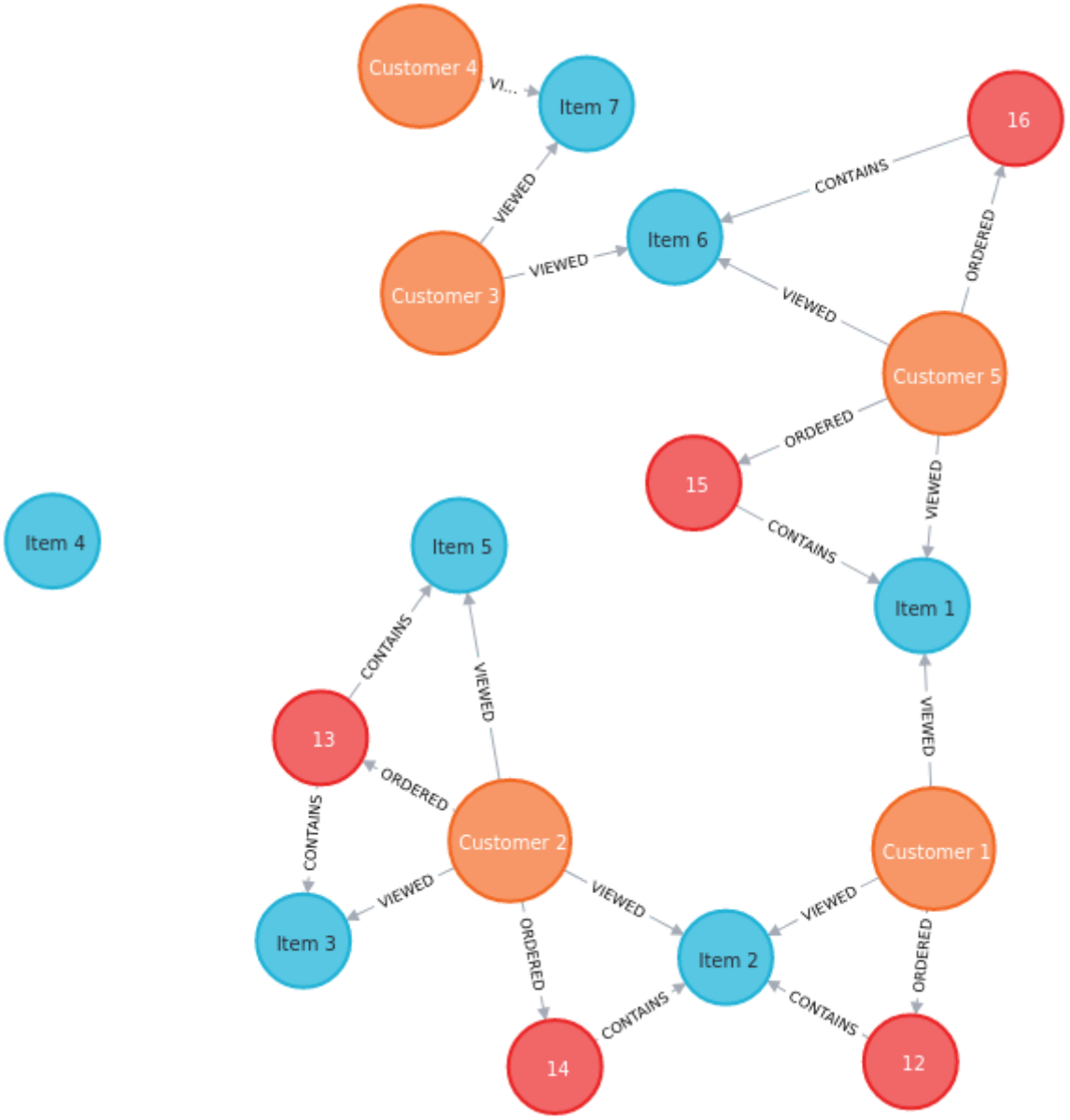
Код ініціалізації бази:

```
CREATE
// customers
(c1:Customers {name: 'Customer 1'}),
(c2:Customers {name: 'Customer 2'}),
(c3:Customers {name: 'Customer 3'}),
(c4:Customers {name: 'Customer 4'}),
(c5:Customers {name: 'Customer 5'}),
// items
(i1:Items {title: 'Item 1'}),
(i2:Items {title: 'Item 2'}),
(i3:Items {title: 'Item 3'}),
(i4:Items {title: 'Item 4'}),
(i5:Items {title: 'Item 5'}),
```

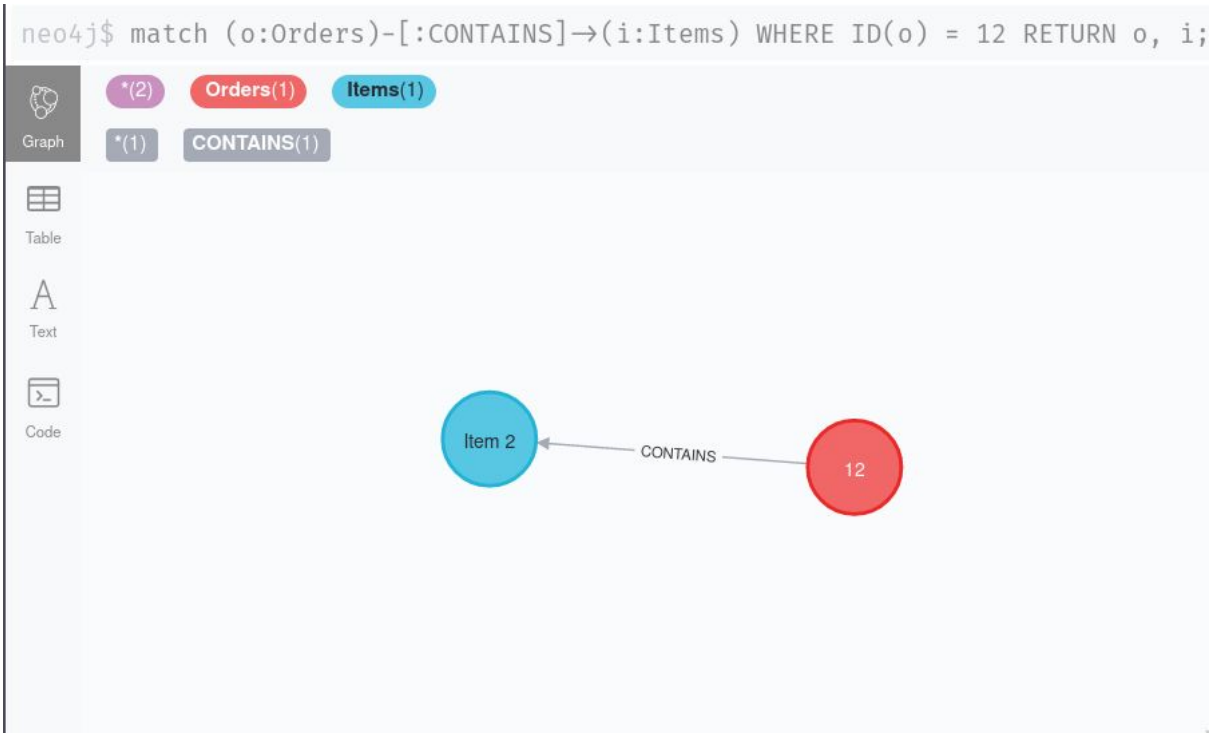
```
(i6:Items {title: 'Item 6'}),
(i7:Items {title: 'Item 7'}),
// orders
// We will use neo4j node ID as unique identifier.
// Maybe it is bad, maybe not. I don't know.
(o1:Orders),
(o2:Orders),
(o3:Orders),
(o4:Orders),
(o5:Orders),
// relationships
// we don't have any requirements to track views, so let view be
relationship.
// At first customer views some items, then order is created, items are
added,
// order is ordered.
(c1)-[:VIEWED]->(i1),
(c1)-[:VIEWED]->(i2),
(o1)-[:CONTAINS]->(i2),
(c1)-[:ORDERED]->(o1),
(c2)-[:VIEWED]->(i5),
(c2)-[:VIEWED]->(i2),
(c2)-[:VIEWED]->(i3),
(o2)-[:CONTAINS]->(i3),
(o2)-[:CONTAINS]->(i5),
(c2)-[:ORDERED]->(o2),
(o3)-[:CONTAINS]->(i2),
(c2)-[:ORDERED]->(o3),
(c3)-[:VIEWED]->(i6),
(c3)-[:VIEWED]->(i7),
(c4)-[:VIEWED]->(i7),
(c5)-[:VIEWED]->(i1),
(o4)-[:CONTAINS]->(i1),
(c5)-[:ORDERED]->(o4),
(c5)-[:VIEWED]->(i6),
(o5)-[:CONTAINS]->(i6),
(c5)-[:ORDERED]->(o5);

// I'd like to add some constraints, but they are in paid version of
neo4j.
```

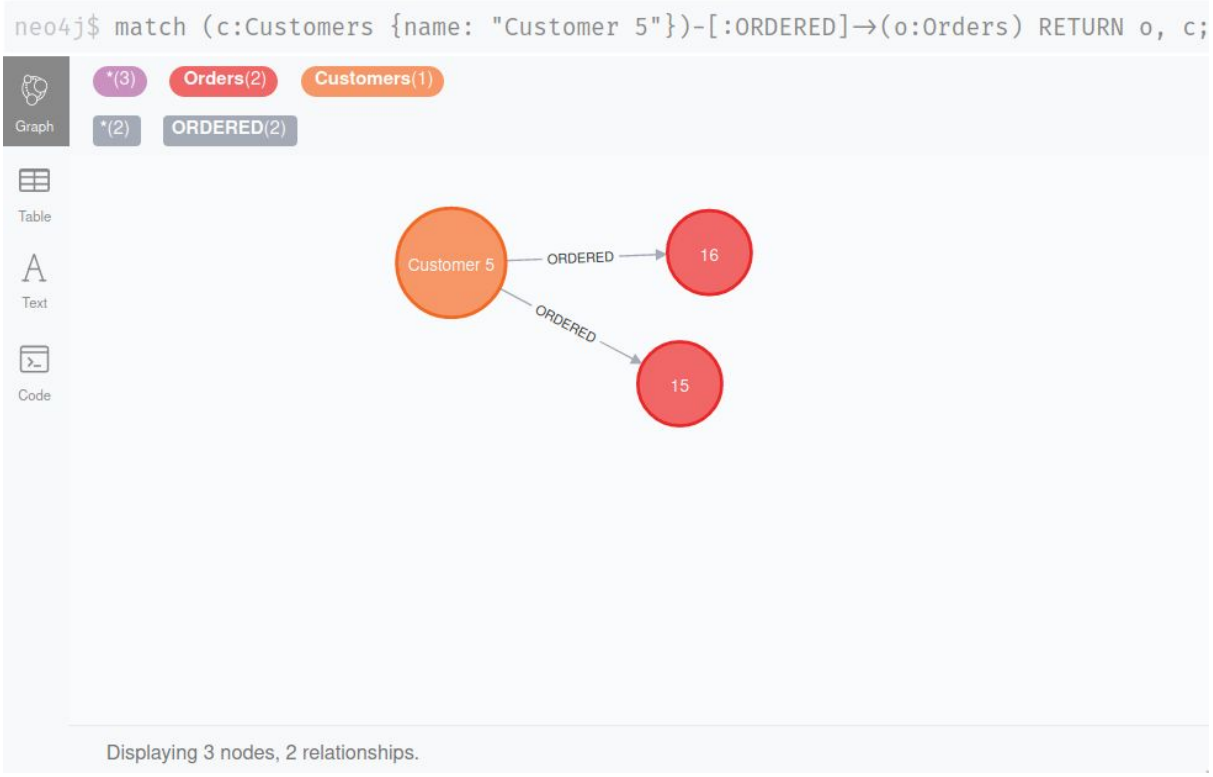
Візуальне представлення графу:
(червоні кружечки це ноди Order, відображаються ID ноди. o1 має 12, o2 - 13, так далі).



Знайти Items які входять в конкретний Order

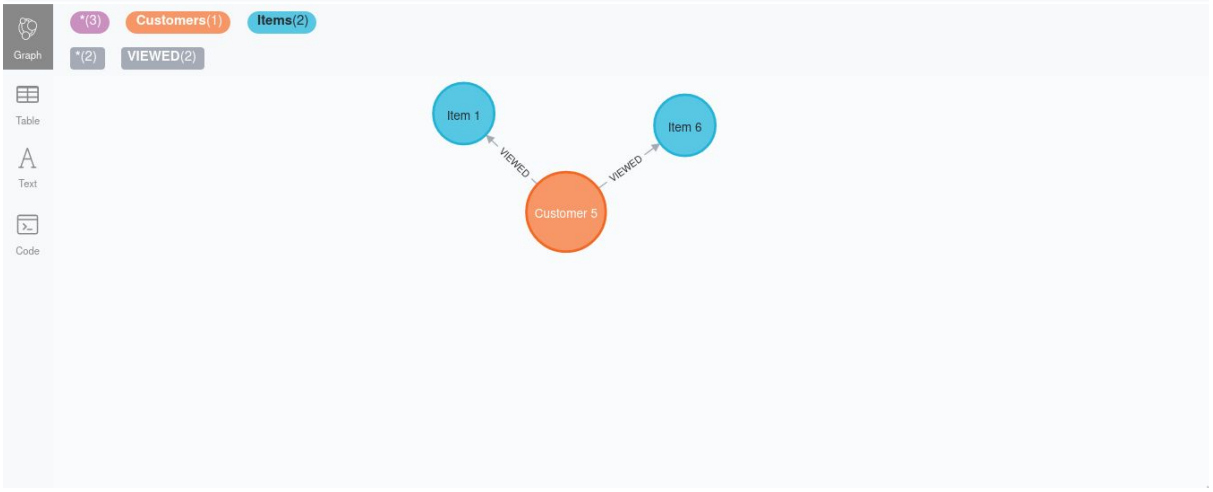


Знайти всі Orders конкретного Customer



Знайти всі Items куплені конкретним Customer

```
neo4j$ match (c:Customers {name: "Customer 5"})-[:ORDERED]→(:Orders)-[:CONTAINS]→(i:Items) RETURN distinct c, i;
```



Знайти кількість Items куплені конкретним Customer з усіх Orders

```
neo4j$ match (c:Customers {name: "Customer 5"})-[:ORDERED]→(:Orders)-[:CONTAINS]→(i:Items) RETURN c.name, count(i);
```

Table | Text | Code

	c.name	count(i)
1	"Customer 5"	2

Started streaming 1 records after 12 ms and completed after 13 ms.

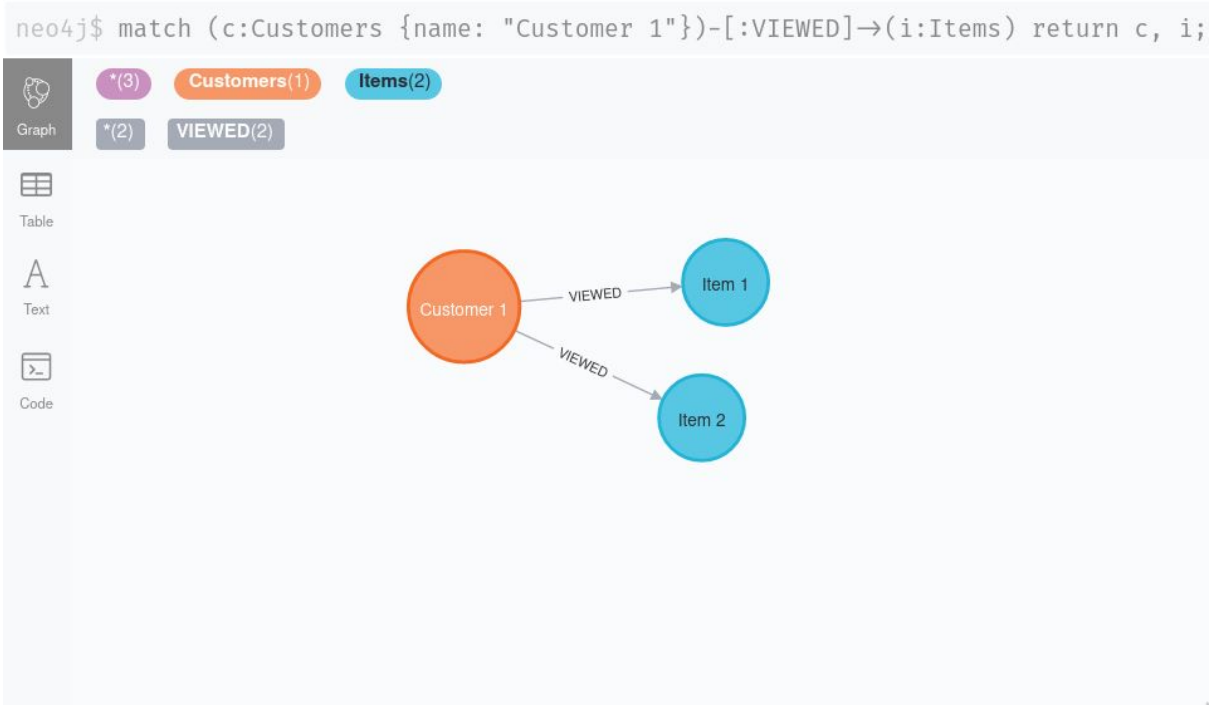
Знайті скільки разів кожен товар був придбаний

```
neo4j$ match(i:Items) optional match (:Orders)-[c:CONTAINS]→(i) return i, count(c);
```

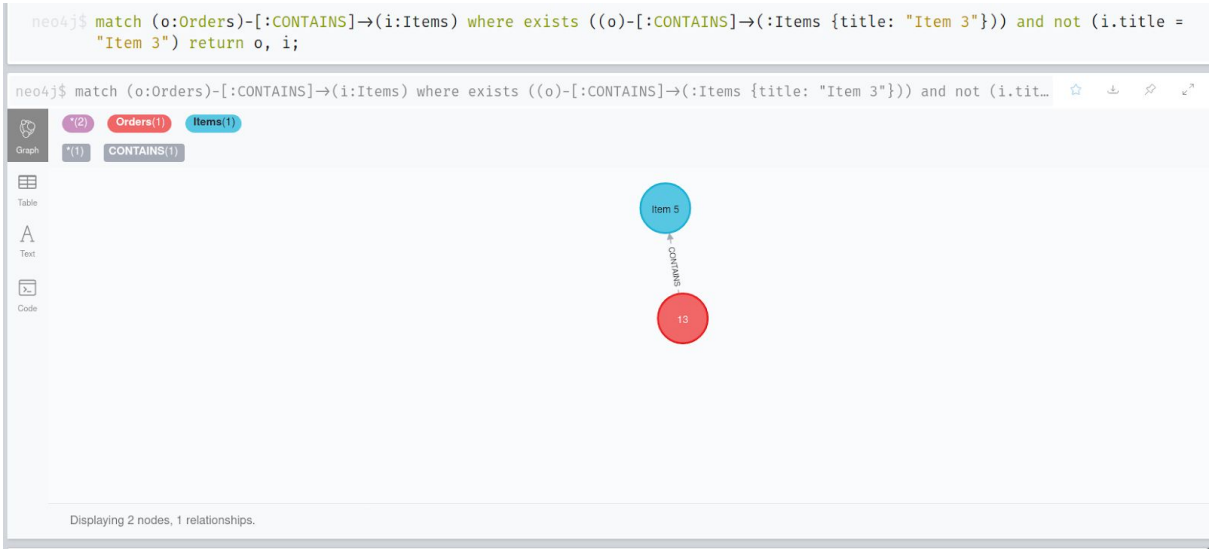
Graph | Table | Text | Code

"i"	"count(c)"
{"title":"Item 1"}	1
{"title":"Item 2"}	2
{"title":"Item 3"}	1
{"title":"Item 4"}	0
{"title":"Item 5"}	1
{"title":"Item 6"}	1
{"title":"Item 7"}	0

Знайти всі Items переглянуті (view) конкретним Customer

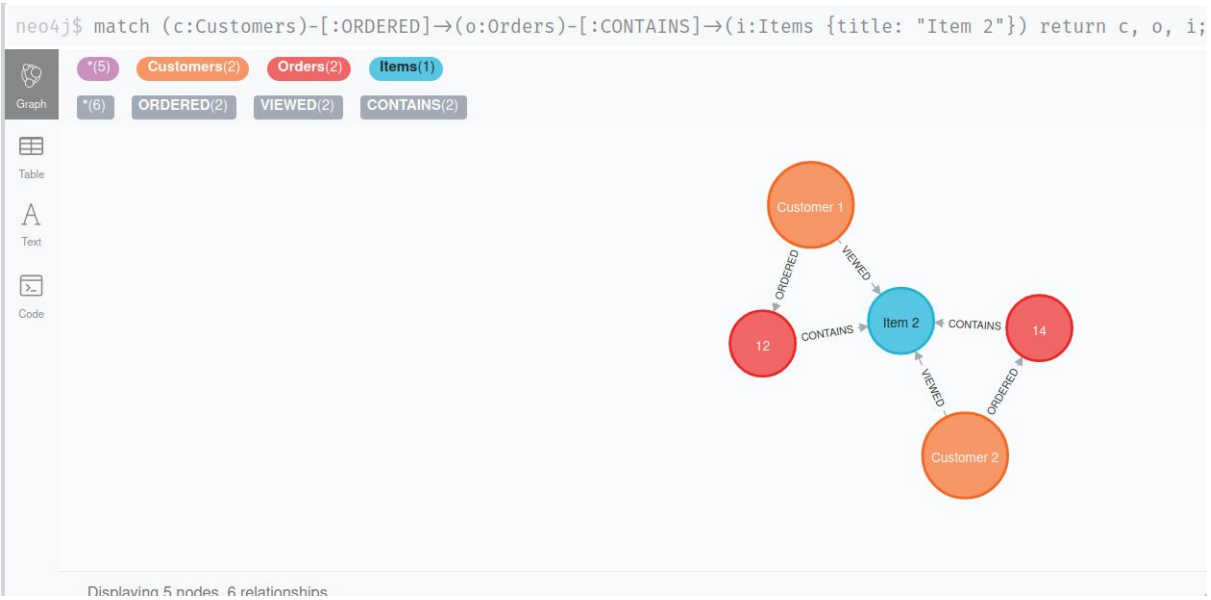


Знайти Items що куплені разом з конкретним Item (тобто все Items що входять до Order разом з даними Item)



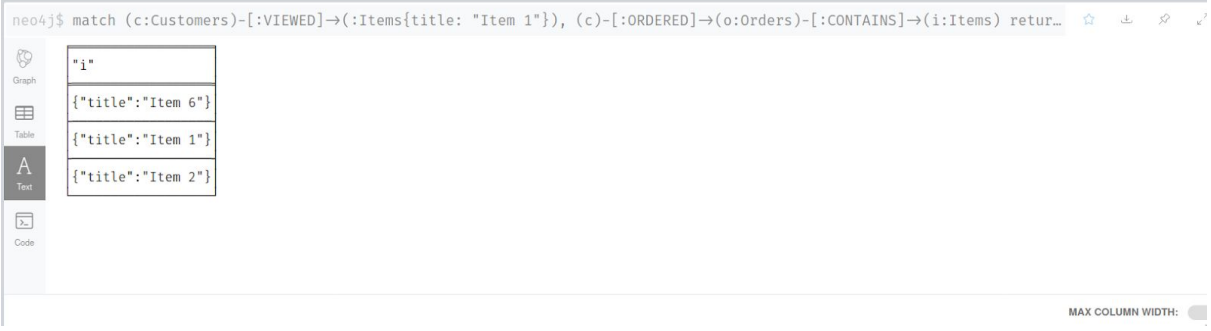
Знайти Customers які купили даний конкретний Item

```
neo4j$ match (c:Customers)-[:ORDERED]→(o:Orders)-[:CONTAINS]→(i:Items {title: "Item 2"}) return c, o, i;
```



Знайти всі Items куплені Customer(s) який переглядав даний конкретний Item

```
neo4j$ match (c:Customers)-[:VIEWED]→(:Items{title: "Item 1"}), (c)-[:ORDERED]→(o:Orders)-[:CONTAINS]→(i:Items) return i;
```



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

```
neo4j$ match (c:Customers{name: "Customer 1"})-[:VIEWED]→(i:Items) where not exists ((c)-[:ORDERED]→(:Orders)-
[:CONTAINS]→(i)) return i;
```