Було створено два .cypher файли (для ініціювання та видалення бд) та один .graphql, але так сталося, що з новими версіями neo4j graphql вже більше не працює, тому даний файл використовувався як опора на створення .json файлу із даними

Init_DB.cypher:

```
create constraint on (i:Item) assert i.id is unique;
create constraint on (c:Customer) assert c.id is unique;
create constraint on (o:Order) assert o.id is unique;
create index on :Customer(name);

call apoc.load.jsonArray('file:///C:/lab3-neo4j/customers.json') yield value
with apoc.convert.toMap(value) as data
with apoc.map.clean(data, [],[''],[],null]) as data
MERGE (c:Customer {id:data.id})
SET
c.name = data.name
FOREACH (id in data.orders | MERGE (o:Order {id:id}) MERGE (c)-[:BUYS]->(o))
FOREACH (id in data.items | MERGE (i:Item {id:id}) MERGE (c)-[:VIEW]->(i))
return c.id, c.name;

call apoc.load.jsonArray('file:///C:/lab3-neo4j/items.json') yield value
with apoc.convert.toMap(value) as data
with apoc.map.clean(data, [],[''],[],null]) as data
MERGE (i:Item {id:data.id})
SET
i.title = data.title
SET
i.price = data.price
return i.id, i.title;

call apoc.load.jsonArray('file:///C:/lab3-neo4j/orders.json') yield value
with apoc.convert.toMap(value) as data
with apoc.map.clean(data, [],[''],[],null]) as data
MERGE (o:Order {id:data.id})
FOREACH (id in data.items | MERGE (i:Item {id:id}) MERGE (o)-[:CONTAIN]->(i))
return o.id;
```

Clean_DB.cypher:

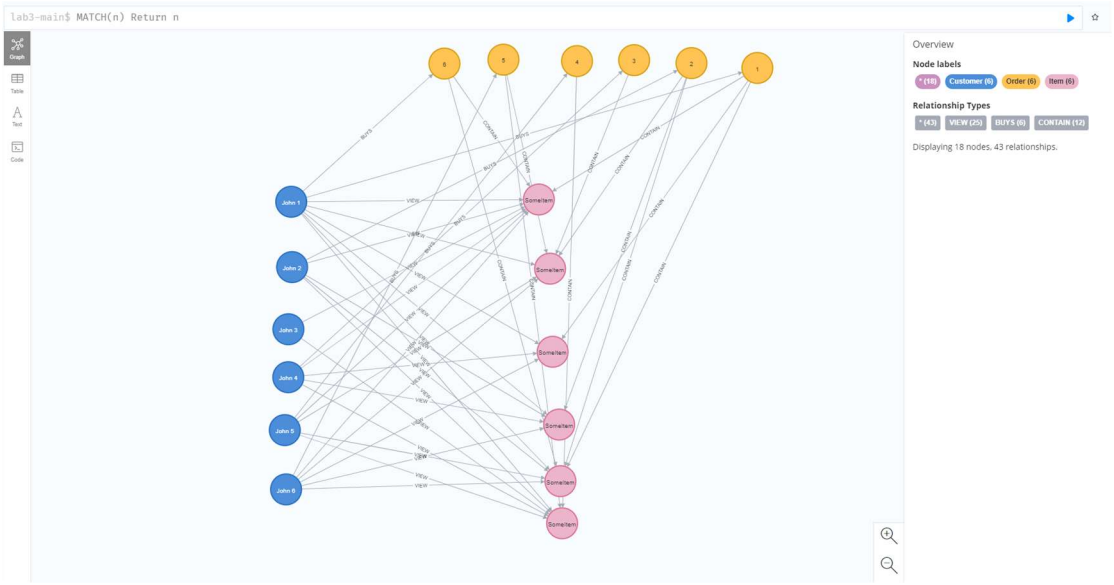
```
drop constraint on (i:Item) assert i.id is unique;
drop constraint on (c:Customer) assert c.id is unique;
drop constraint on (o:Order) assert o.id is unique;
drop index on :Customer(name);

MATCH (everything) DETACH DELETE everything;
```

Schema.graphql:

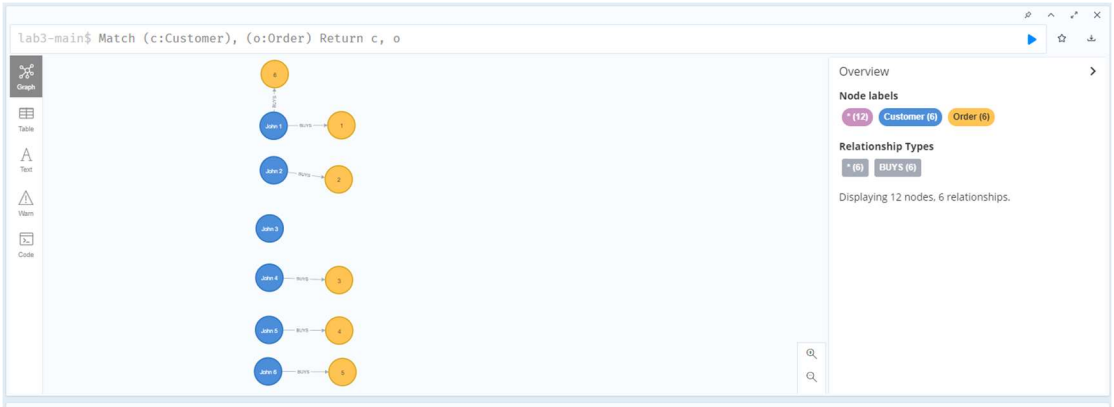
```
type Order {
        id: ID!
        items: [Item] @relation(name:"CONTAINS", direction:OUT)
        customer: Customer @relation(name:"BUYS", direction:IN)
}
type Item {
    id: ID!
    title: String!
    price: Int!
    orders: [Order] @relation(name:"CONTAINS", direction:IN)
    customers: [Customer] @relation(name:"VIEW", direction:IN)
}

type Customer {
    id: ID!
    name: String!
    orders: [Order] @relation(name:"BUYS", direction:OUT)
    items: [Item] @relation(name:"VIEW", direction:OUT)
}
```

Змоделювати наступну предметну область:

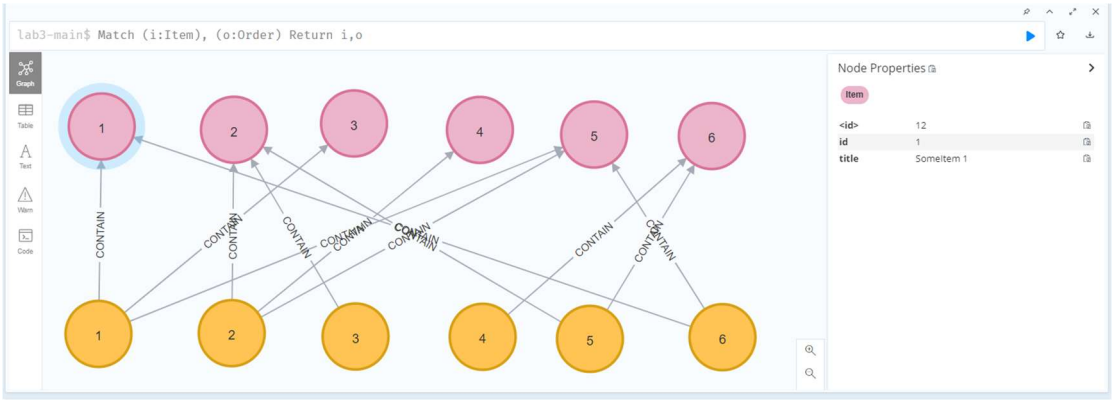
- Є: Items, Customers, Orders
- Customer може додати Item(s) до Order (тобто купити Товар)



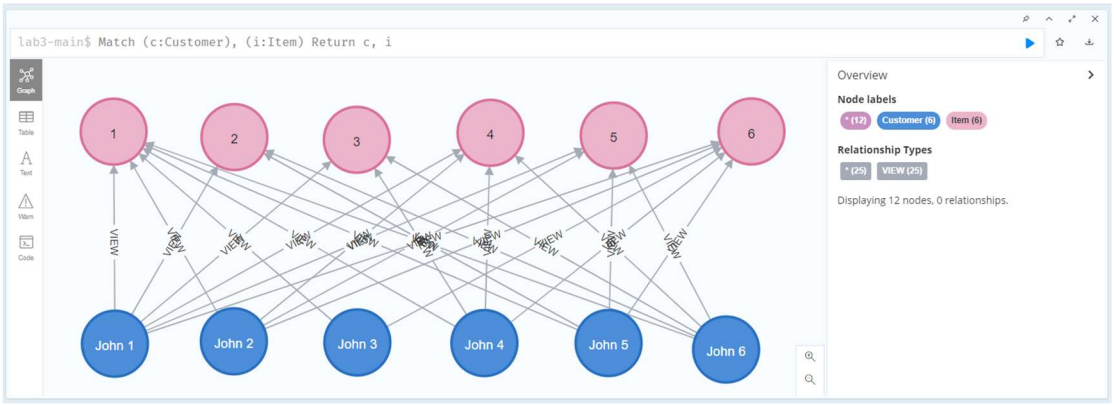
- У Customer може бути багато Orders



- Item може входити в багато Orders, і у Item є вартість

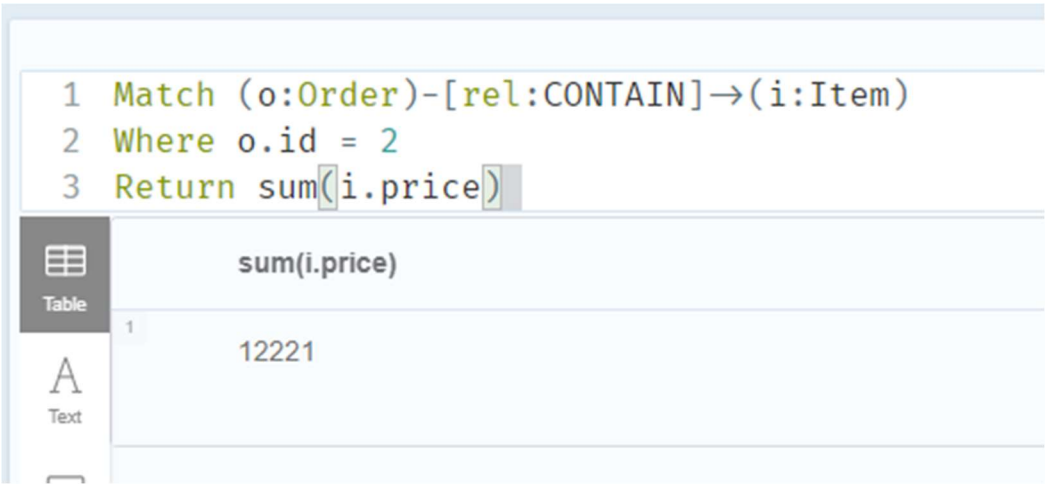


- Customer може переглядати (view), але при цьому не купувати Items



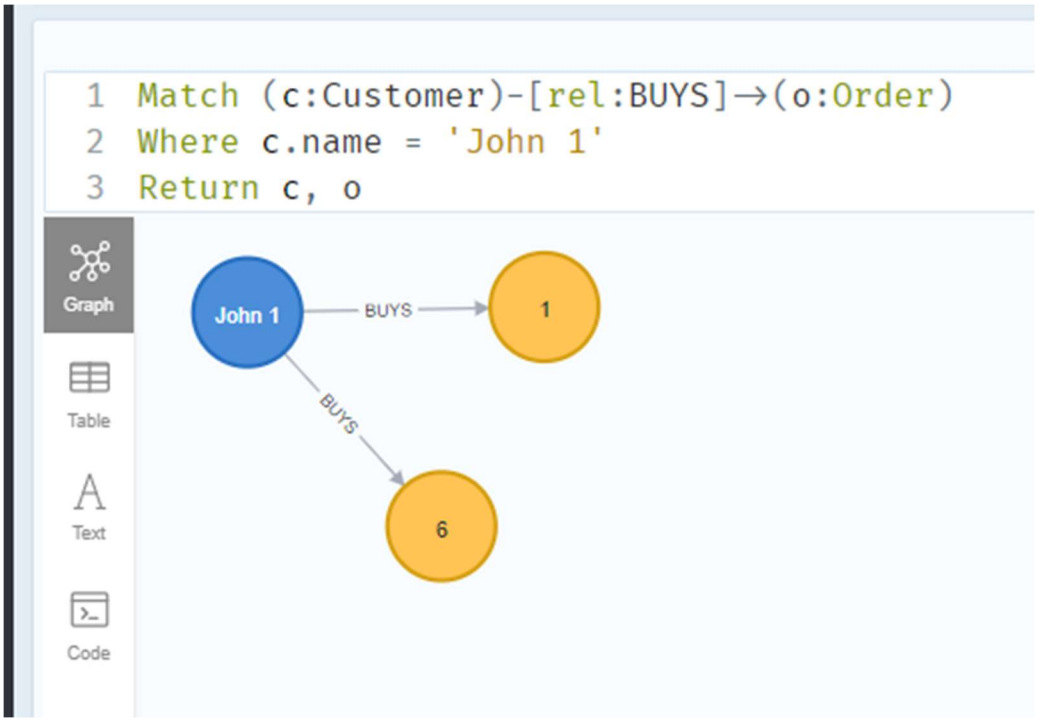
Знайти Items які входять в конкретний Order

Підрахувати вартість конкретного Order



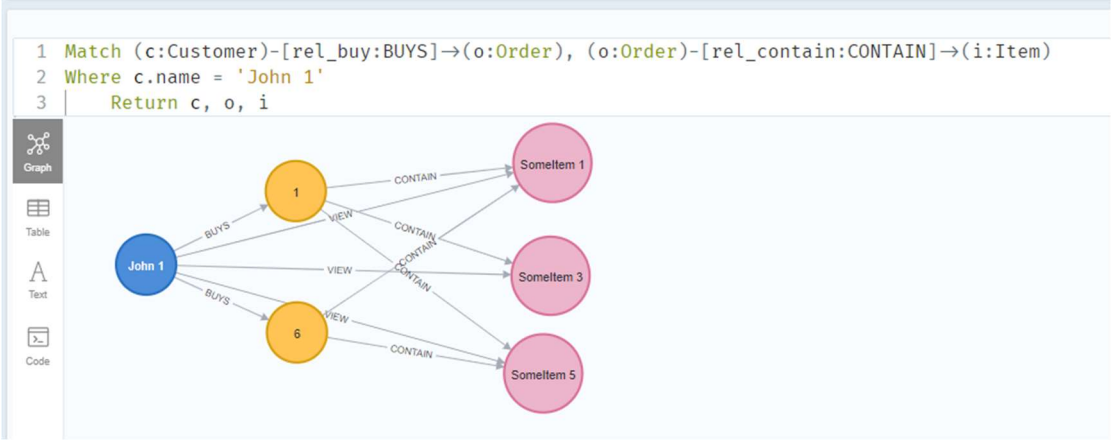
Знайти всі Orders конкретного Customer

```
Match (c:Customer)-[rel:BUYS]→(o:Order)
Where c.name = 'John 1'
Return c, o
```



Знайти всі Items куплені конкретним Customer (через Order)

```
Match (c:Customer)-[rel_buy:BUYS]→(o:Order), (o:Order)-[rel_contain:CONTAIN]→(i:Item)
Where c.name = 'John 1'
   Return c, o, i
```



Знайти кількість Items куплені конкретним Customer (через Order)

```
Match (c:Customer)-[rel_buys:BUYS]→(o:Order), (o:Order)-[rel_contain:CONTAIN]→(i:Item)
Where c.name = "John 1"
Return count(i)
```

Table

	count(i)
1	5

Text

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

```
Match (c:Customer)-[rel_buy:BUYS]→(o:Order), (o:Order)-[rel_contain:CONTAIN]→(i:Item)
Where c.name = "John 1"
Return sum(i.price)
```

Table

	sum(i.price)
1	16665

Text

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

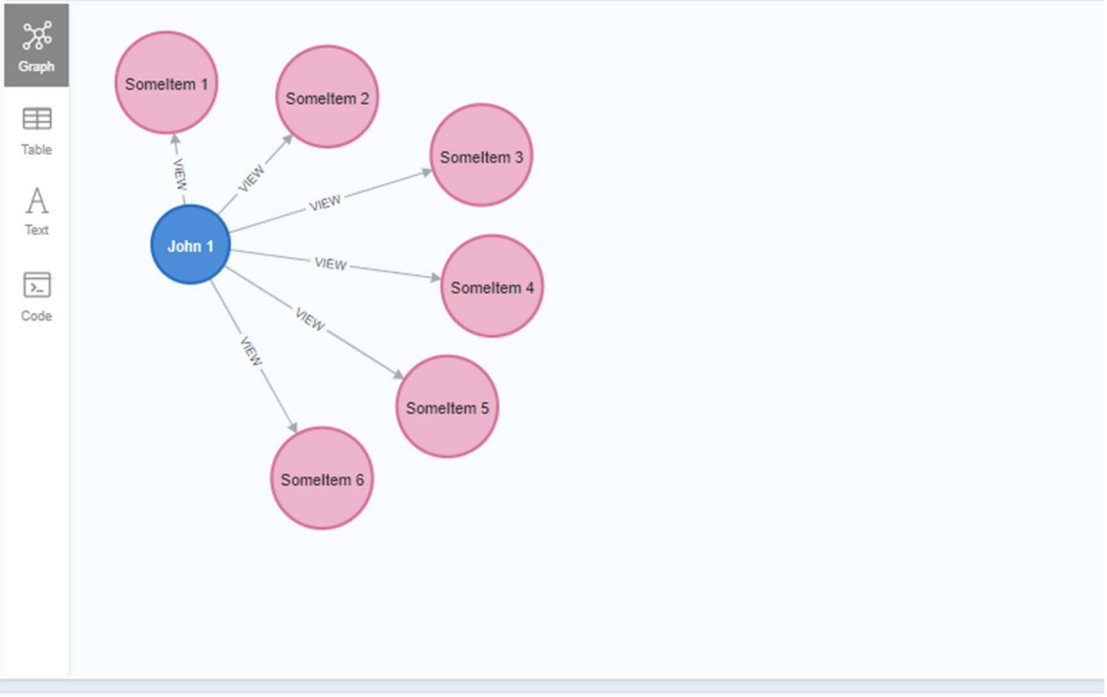
```
Match (o:Order)-[rel_contain:CONTAIN]→(i:Item)
Return i.title,
Count (rel_contain)
Order By Count(rel_contain) Desc
```

Table | Text | Code

	i.title	Count (rel_contain)
1	"SomeItem 2"	3
2	"SomeItem 5"	3
3	"SomeItem 1"	2
4	"SomeItem 6"	2
5	"SomeItem 3"	1
6	"SomeItem 4"	1

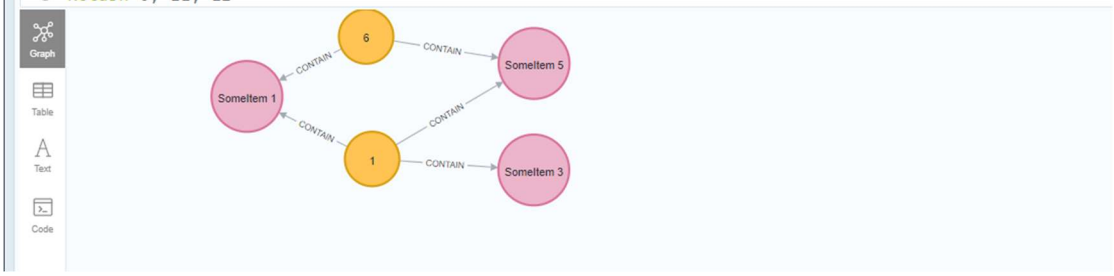
Знайти всі Items переглянуті (view) конкретним Customer

```
Match (c:Customer)-[rel_view:VIEW]→(i:Item)
Where c.name = "John 1"
Return c, i
```



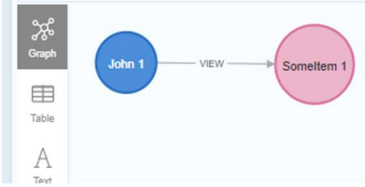
Знайти інші Items що купувались разом з конкретним Item (тобто всі Items що входять до Order-s разом з даними Item)

```
Match (o:Order)-[rel_contain1:CONTAIN]→(i1:Item), (o:Order)-[rel_contain2:CONTAIN]→(i2:Item)
Where i1.title = "SomeItem 1"
Return o, i1, i2
```

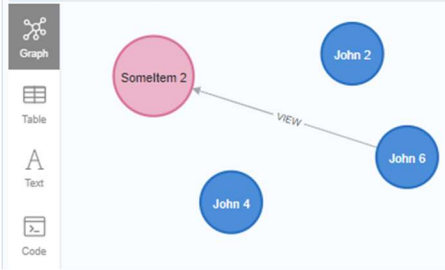


Знайти Customers які купили даний конкретний Item

```
Match (o:Order)-[rel_contain:CONTAIN]→(i:Item), (o:Order)←[rel_buy:BUYS]-(c:Customer)
Where i.title = "SomeItem 1"
Return i, c
```



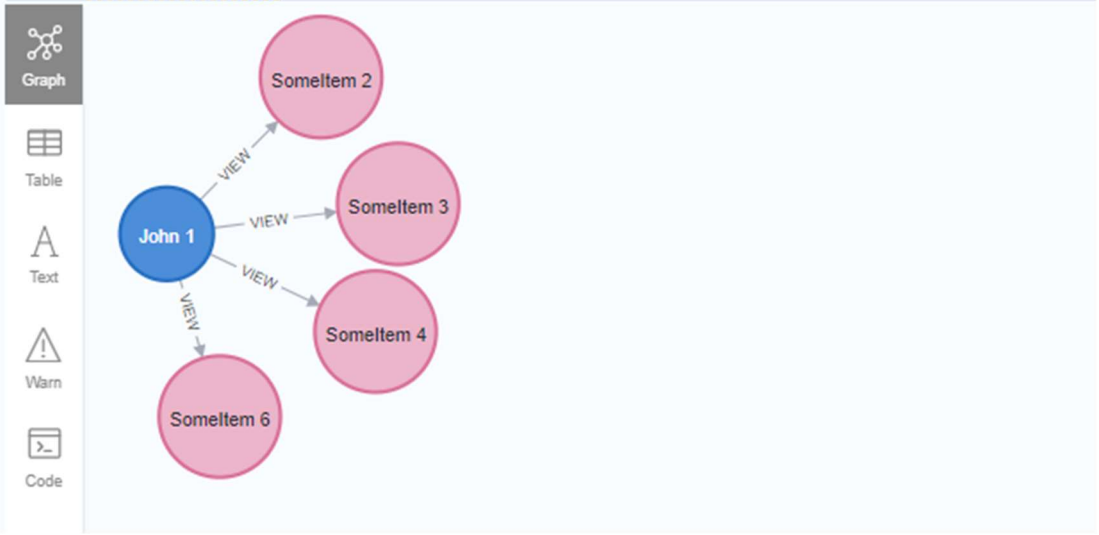

```
Match (o:Order)-[rel_contain:CONTAIN]→(i:Item), (o:Order)←[rel_buy:BUYS]-(c:Customer)
Where i.title = "SomeItem 2"
Return i, c
```



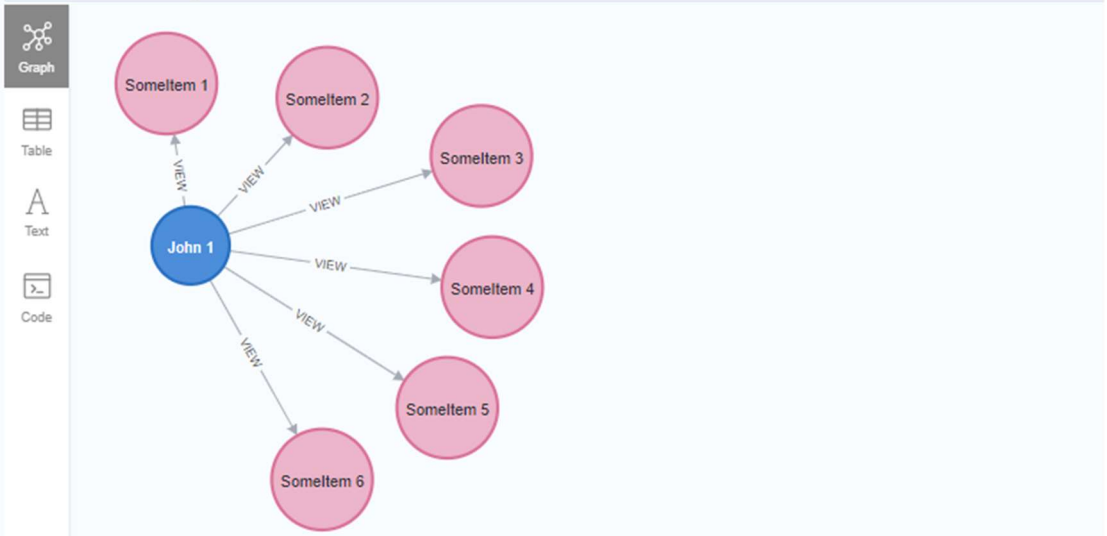
Чому немає зв’язку? Бо магазин надає механіку «сюрпризу»)0)0 (це було зроблено для теста, як поведе себе БД, якщо будуть такі косвені зв’язки)

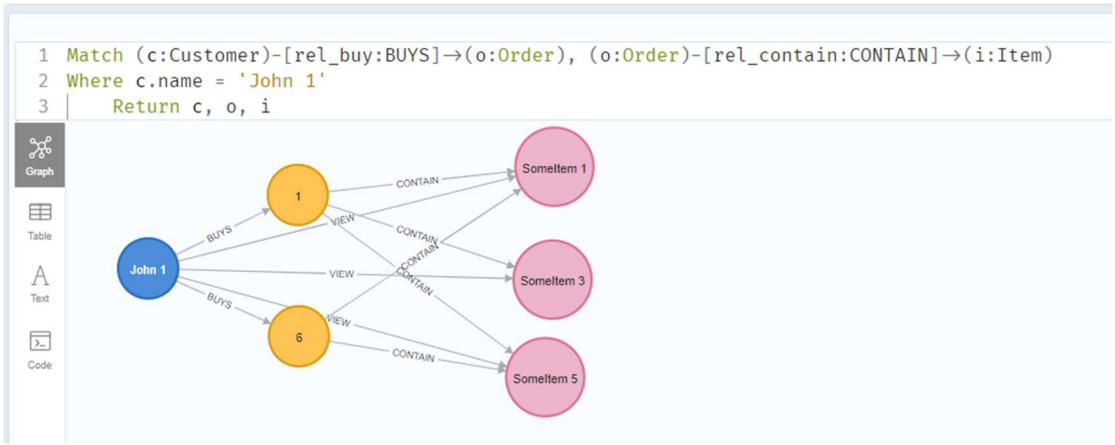
Знайти для певного Customer(a) товари, які він переглядав, але не купив

```
Match (c:Customer)-[rel_view:VIEW]→(i:Item),
(c)-[rel_buys:BUYS]→(o:Order)
WHERE c.name = "John 1"
and not (o)-[:CONTAIN]→(i)
Return c, i
```




```
Match (c:Customer)-[rel_view:VIEW]→(i:Item)
Where c.name = "John 1"
Return c, i
```





Json datasets:

Customers.json:


```
[
	{
		"id": 1,
		"name": "John 1",
		"orders": [1, 6],
		"items": [1, 2, 3, 4, 5, 6]
	},
	{
		"id": 2,
		"name": "John 2",
		"orders": [2],
		"items": [1, 4, 5, 6]
	},
	{
		"id": 3,
		"name": "John 3",
		"orders": [],
		"items": [1, 6]
	},
	{
		"id": 4,
		"name": "John 4",
		"orders": [3],
		"items": [1, 3, 4, 6]
	},
	{
		"id": 5,
		"name": "John 5",
		"orders": [4],
		"items": [1, 2, 5, 6]
	},
	{
		"id": 6,
		"name": "John 6",
		"orders": [5],
		"items": [1, 2, 3, 4, 5]
	}
]
```


Items.json:


```
[
	{
		"id": 1,
		"title": "SomeItem 1",
		"price": 1111
	},
	{
		"id": 2,
		"title": "SomeItem 2",
		"price": 2222
	},
	{
		"id": 3,
		"title": "SomeItem 3",
		"price": 3333
	},
	{
		"id": 4,
		"title": "SomeItem 4",
```

```
        "price": 4444
    },
    {
        "id": 5,
        "title": "SomeItem 5",
        "price": 5555
    },
    {
        "id": 6,
        "title": "SomeItem 6",
        "price": 6666
    }
]
```


Orders.json:


```
[
    {
        "id": 1,
        "items": [1, 3, 5]
    },
    {
        "id": 2,
        "items": [2, 5, 4]
    },
    {
        "id": 3,
        "items": [2]
    },
    {
        "id": 4,
        "items": [6]
    },
    {
        "id": 5,
        "items": [2, 6]
    },
    {
        "id": 6,
        "items": [1, 5]
    }
]
```