Міністерство освіти і науки України Національний технічний університет України «Київський політехнічний інститут ім. І. Сікорського » Фізико-технічний інститут

Проектування високонавантажених систем

Лабораторна робота №3

Робота з базовими функціями граф-орієнтованої БД на прикладі Neo4j

Виконав:

Студент групи ФБ-41мп

Заріцький О. В.

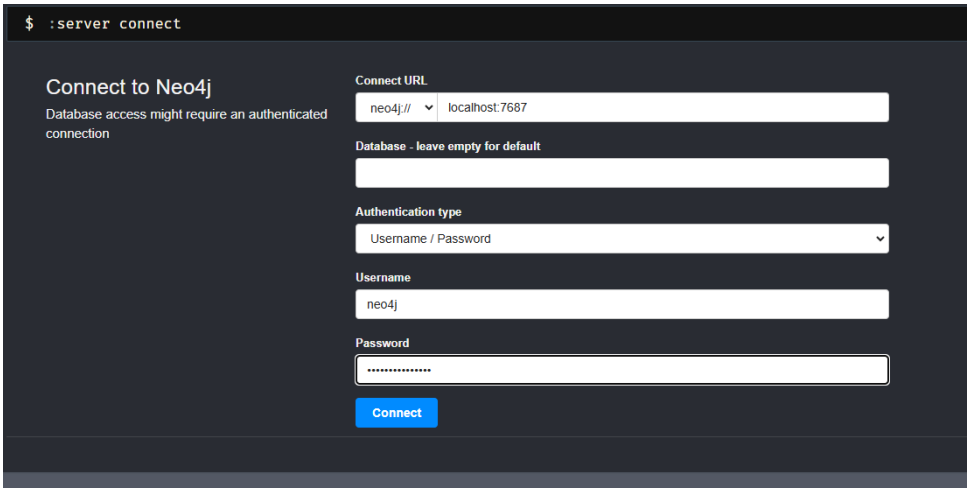
Київ 2024 р.

Встановлення Neo4j

compose.yml

```yaml
services:
  neo4j:
    image: neo4j
    ports:
      - "7474:7474"
      - "7687:7687"
    volumes:
      - ./neo4j/data:/data
      - ./neo4j/logs:/logs
    environment:
      - NEO4J_AUTH=neo4j/zaritsky_fb41mp
      - NEO4J_ACCEPT_LICENSE_AGREEMENT=yes
      - NEO4JLABS_PLUGINS=["graph-data-science", "apoc"]
volumes:
  neo4j:
    driver: local

```



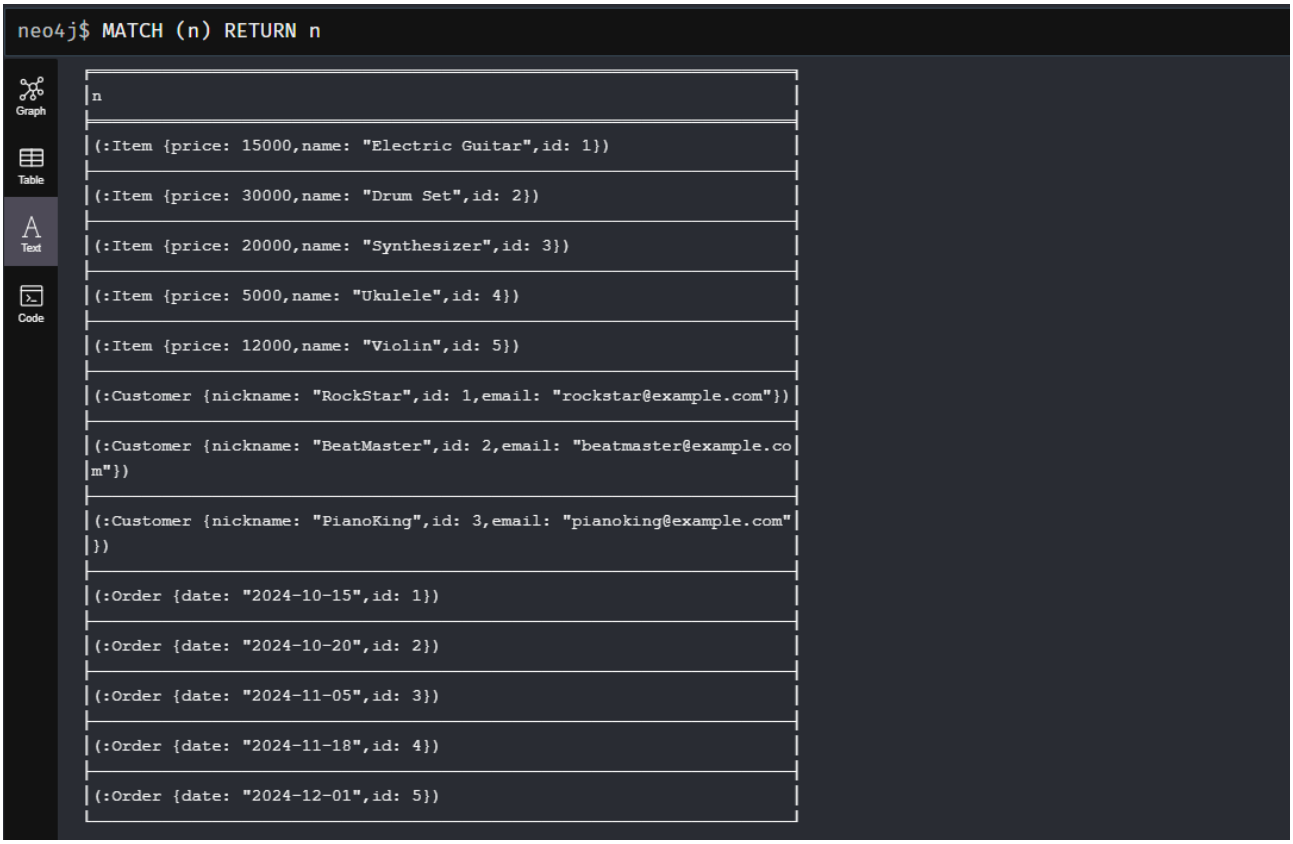
Змоделювати наступну предметну область:

Є: Items, Customers, Orders

```
CREATE (i1:Item {id: 1, name: 'Electric Guitar', price: 15000})
CREATE (i2:Item {id: 2, name: 'Drum Set', price: 30000})
CREATE (i3:Item {id: 3, name: 'Synthesizer', price: 20000})
CREATE (i4:Item {id: 4, name: 'Ukulele', price: 5000})
CREATE (i5:Item {id: 5, name: 'Violin', price: 12000})

CREATE (c1:Customer {id: 1, nickname: 'RockStar', email: 'rockstar@example.com'})
CREATE (c2:Customer {id: 2, nickname: 'BeatMaster', email: 'beatmaster@example.com'})
CREATE (c3:Customer {id: 3, nickname: 'PianoKing', email: 'pianoking@example.com'})

CREATE (o1:Order {id: 1, date: '2024-10-15'})
CREATE (o2:Order {id: 2, date: '2024-10-20'})
CREATE (o3:Order {id: 3, date: '2024-11-05'})
CREATE (o4:Order {id: 4, date: '2024-11-18'})
CREATE (o5:Order {id: 5, date: '2024-12-01'})
```



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

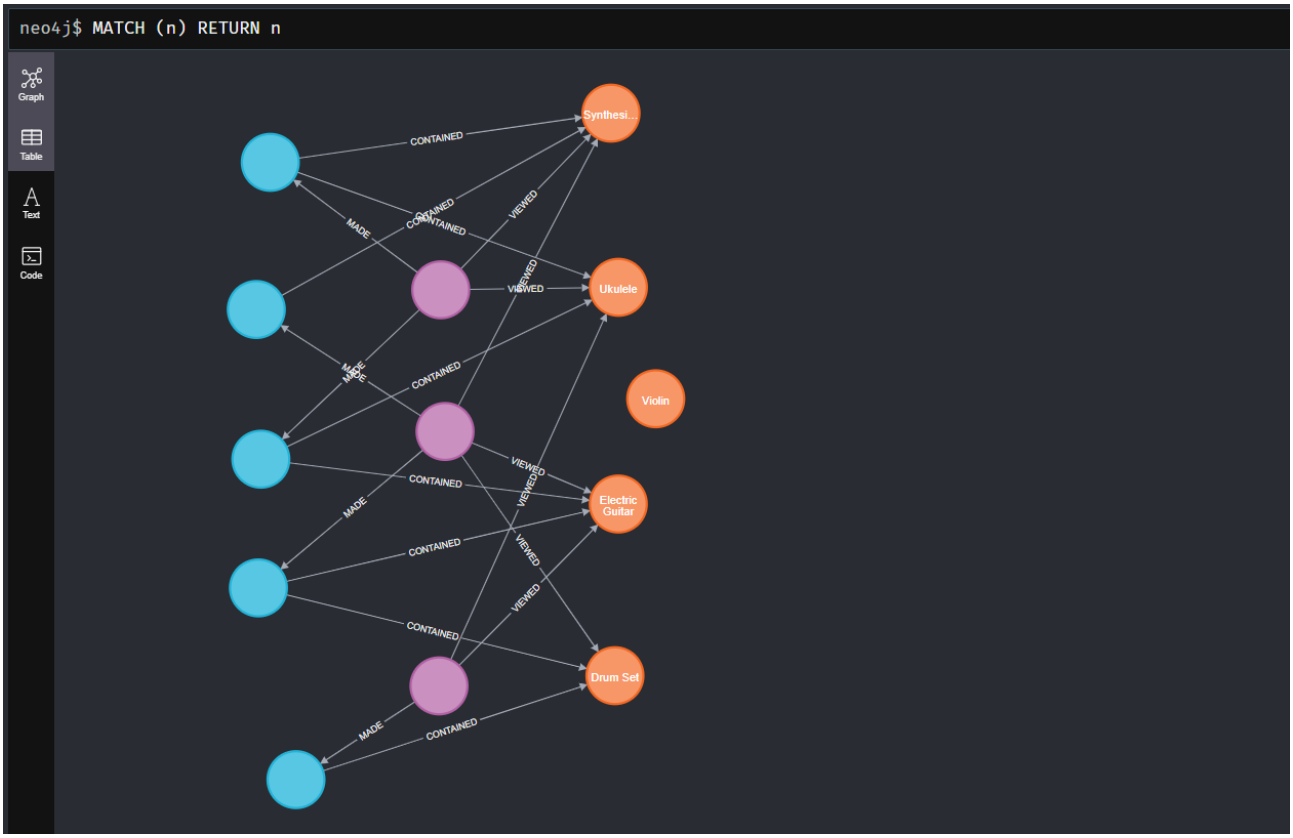
```
// Підбір покупців і замовлень
MATCH (c1:Customer {id: 1}), (c2:Customer {id: 2}), (c3:Customer {id: 3}),
      (o1:Order {id: 1}), (o2:Order {id: 2}), (o3:Order {id: 3}),
      (o4:Order {id: 4}), (o5:Order {id: 5})

CREATE
(c1)-[:MADE]->(o1),
(c1)-[:MADE]->(o2),
(c2)-[:MADE]->(o3),
(c3)-[:MADE]->(o4),
(c2)-[:MADE]->(o5);


// Зв'язки між замовленнями і товарами
MATCH (o1:Order {id: 1}), (o2:Order {id: 2}), (o3:Order {id: 3}),
      (o4:Order {id: 4}), (o5:Order {id: 5}),
      (i1:Item {id: 1}), (i2:Item {id: 2}), (i3:Item {id: 3}), (i4:Item {id: 4})

CREATE
(o1)-[:CONTAINED]->(i1),
(o1)-[:CONTAINED]->(i2),
(o2)-[:CONTAINED]->(i3),
(o3)-[:CONTAINED]->(i4),
(o3)-[:CONTAINED]->(i1),
(o4)-[:CONTAINED]->(i2),
(o5)-[:CONTAINED]->(i3),
(o5)-[:CONTAINED]->(i4);
```

```
// Перегляд товарів
MATCH (c1:Customer {id: 1}), (c2:Customer {id: 2}), (c3:Customer {id: 3}),
      (i1:Item {id: 1}), (i2:Item {id: 2}), (i3:Item {id: 3}), (i4:Item {id: 4})
CREATE
(c1)-[:VIEWED {viewDate: datetime()}]->(i1),
(c1)-[:VIEWED {viewDate: datetime()}]->(i2),
(c1)-[:VIEWED {viewDate: datetime()}]->(i3),
(c2)-[:VIEWED {viewDate: datetime()}]->(i3),
(c2)-[:VIEWED {viewDate: datetime()}]->(i4),
(c3)-[:VIEWED {viewDate: datetime()}]->(i1),
(c3)-[:VIEWED {viewDate: datetime()}]->(i4);
```



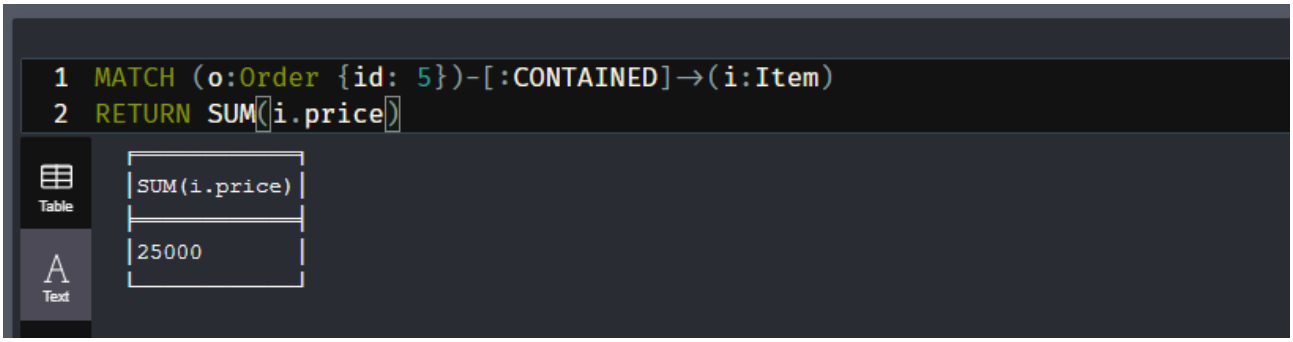
Написати наступні види запитів:

Знайти Items які входять в конкретний Order

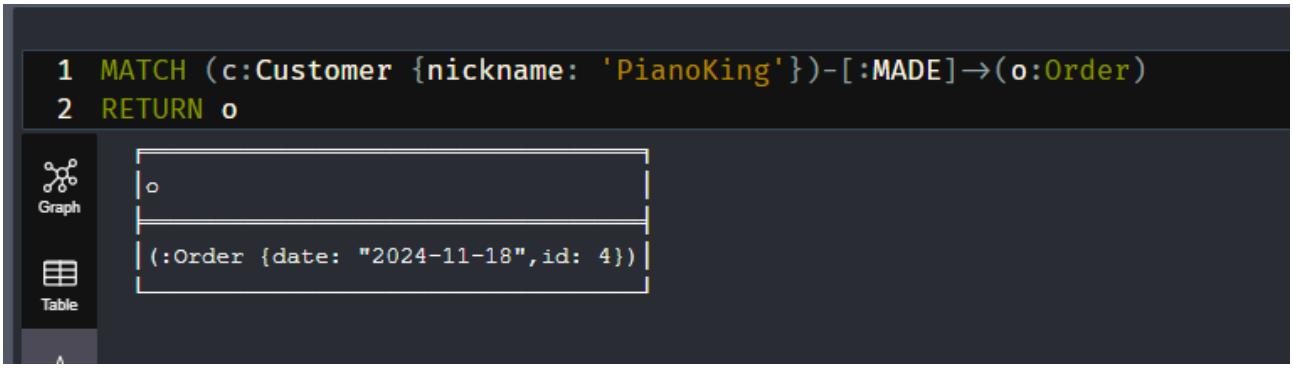
```
1 MATCH (o:Order {id: 5})-[:CONTAINED]→(i:Item)
2 RETURN i
```

i
(:Item {price: 5000,name: "Ukulele",id: 4})
(:Item {price: 20000,name: "Synthesizer",id: 3})

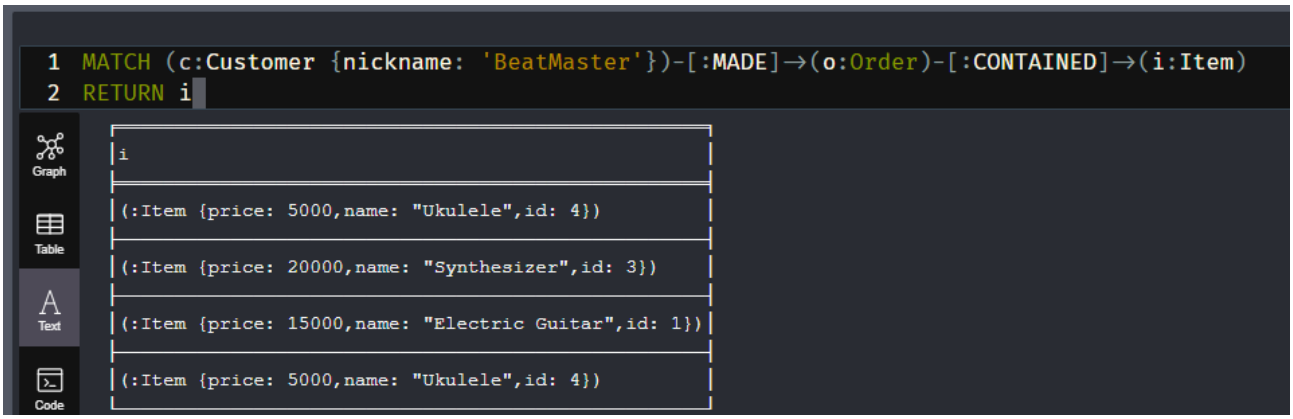
Підрахувати вартість конкретного Order



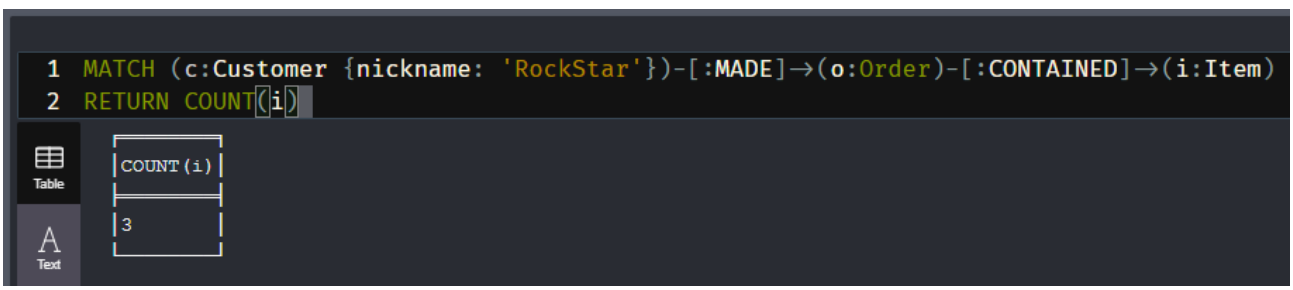
Знайти всі Orders конкретного Customer



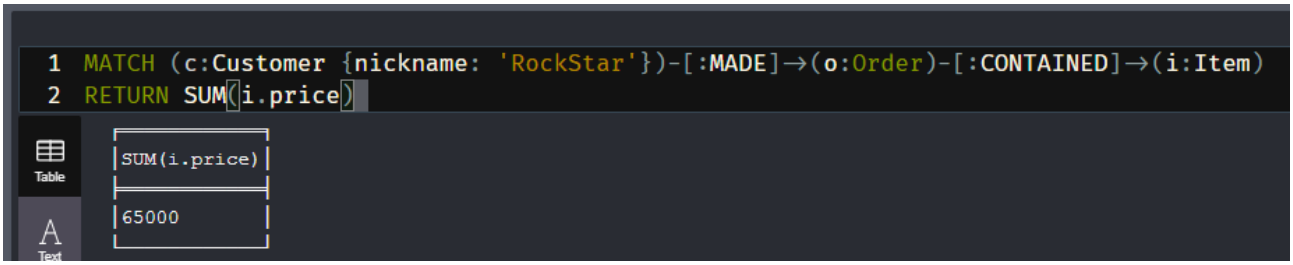
Знайти всі Items куплені конкретним Customer (через Order)



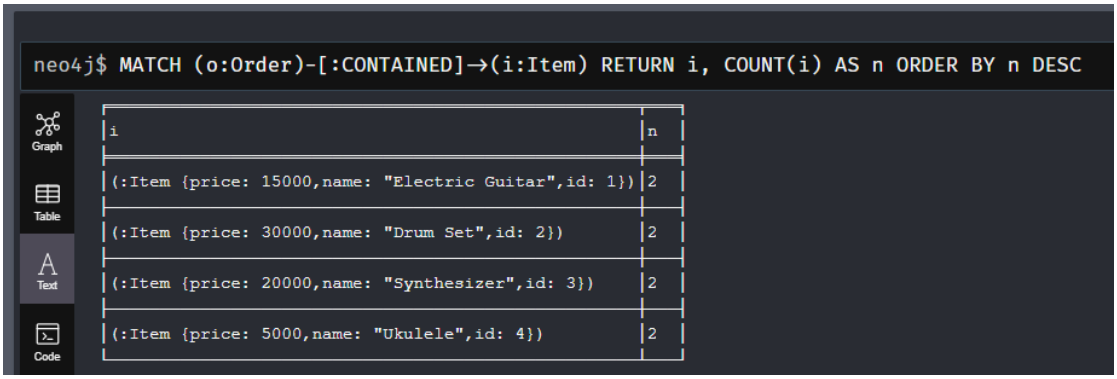
Знайти кількість Items куплені конкретним Customer (через Order)



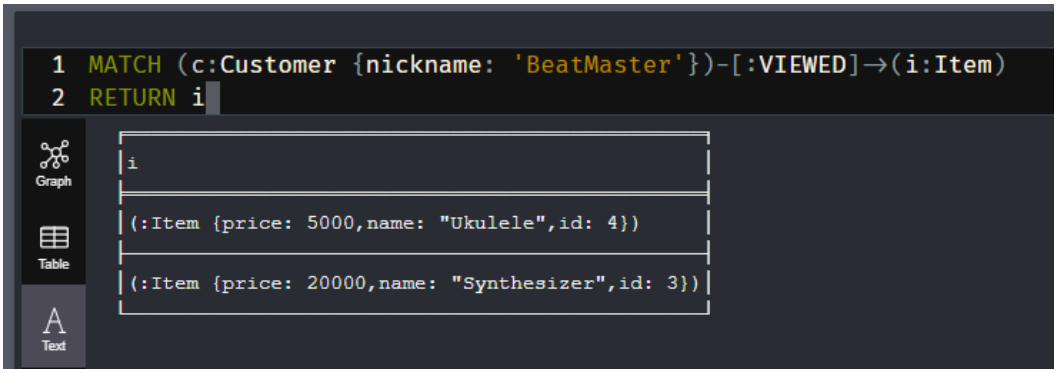
Знайти для Customer на яку суму він придбав товарів (через Order)



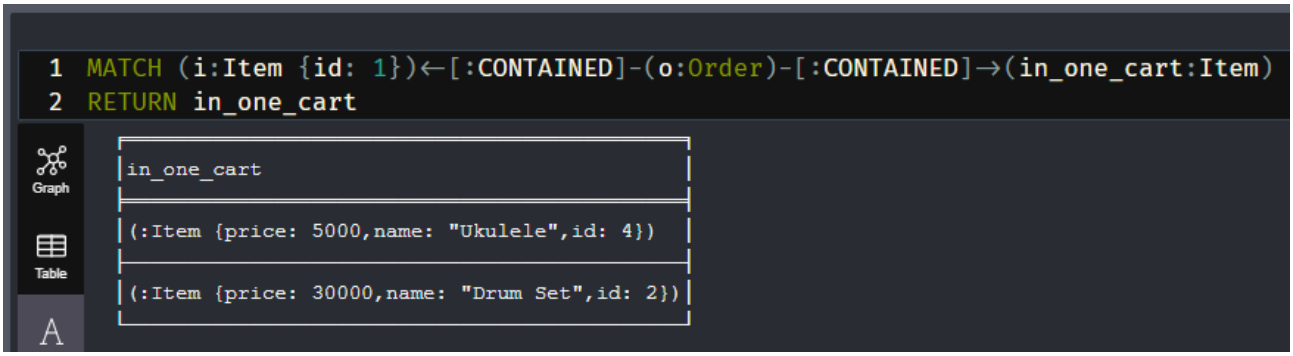
Знайті скільки разів кожен товар був придбаний, відсортувати за цим значенням



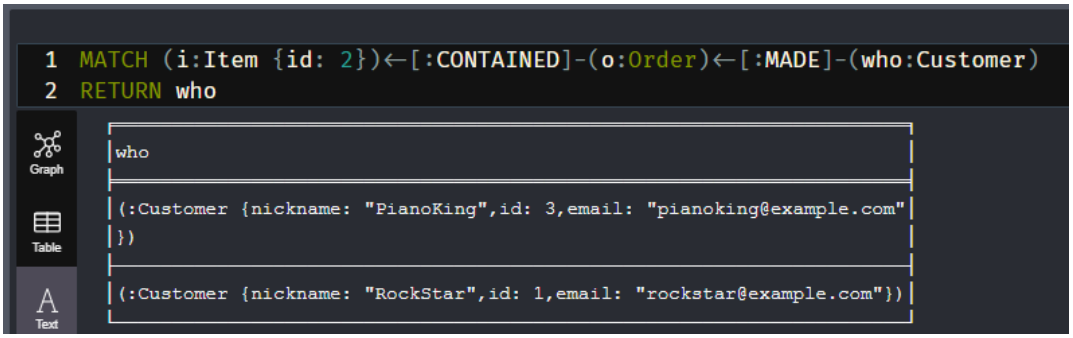
Знайти всі Items переглянуті (view) конкретним Customer



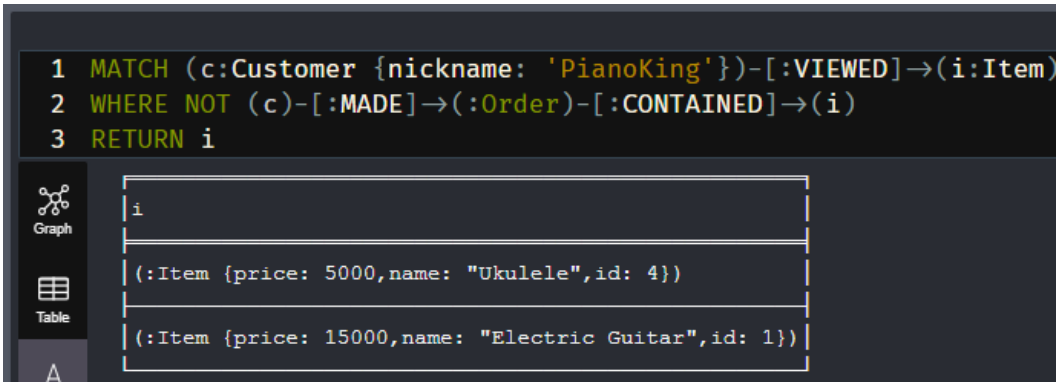
Знайти інші Items що купувались разом з конкретним Item (тобто всі Items що входять до Order-s разом з даними Item)



Знайти Customers які купили даний конкретний Item



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



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

Виконання:

```
uri = "bolt://localhost:7687"
user = "neo4j"
password = "zaritsky_fb41mp"

driver = GraphDatabase.driver(uri, auth=(user, password))

def initialize_likes_field(driver, item_id):
    with driver.session() as session:
        session.run("""
            MATCH (i:Item {id: $item_id})
            SET i.likes = 0
            """, item_id=item_id)

def increment_likes(driver, item_id, increments):
    with driver.session() as session:
        for _ in range(increments):
            session.run("""
                MATCH (i:Item {id: $item_id})
                SET i.likes = i.likes + 1
                """, item_id=item_id)

def get_likes(driver, item_id):
    with driver.session() as session:
        result = session.run("""
            MATCH (i:Item {id: $item_id})
            RETURN i.likes AS likes
            """, item_id=item_id)
        return result.single()["likes"]
```

```python
def safe_increment_likes(driver, item_id, increments):
    retries = 5
    for _ in range(increments):
        success = False
        attempts = 0
        while not success and attempts < retries:
            try:
                increment_likes(driver, item_id, 1)
                success = True
            except TransientError:
                attempts += 1
                time.sleep(0.1)
```

Результат:

```
Initializing 'likes' for the item with ID: 1
Starting threads for incrementing...
Final 'likes' count for item with ID 1: 100000
Execution time: 158.51 seconds
```