НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Проектування високонавантажених систем

Лабораторна робота №4

Виконав:
Поночевний Назар ФІ-92
Перевірив:
Родіонов А. М.

Київ – 2022

ЛАБОРАТОРНА РОБОТА №4

Робота з базовими функціями граф-орієнтованої БД на прикладі Neo4j

Завдання:

Змоделювати наступну предметну область:

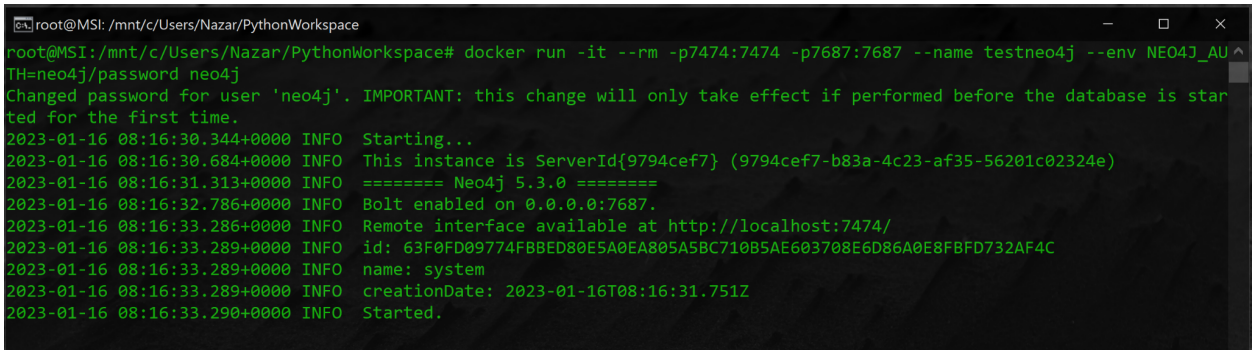
- Є: Items, Customers, Orders
- Customer може додати Item(s) до Order (тобто купити Товар)
- У Customer може бути багато Orders
- Item може входити в багато Orders, і у Item є вартість
- Customer може переглядати (view), але при цьому не купувати Items

Написати наступні види запитів:

- Знайти Items які входять в конкретний Order
- 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

Результати

1) Інсталяція Neo4j



2) Код:

```
from neo4j import GraphDatabase


# find Items that are included in a specific Order
```

```
def find_items_in_order(order_number):
    with driver.session() as session:
        result = session.run("MATCH (i:Item)-[:INCLUDED_IN]->(o:Order
{number: $order_number}) RETURN i", order_number=order_number)
        return result.values()

# calculate the cost of a specific Order
def calculate_cost_of_order(order_number):
    with driver.session() as session:
        result = session.run("MATCH (i:Item)-[:INCLUDED_IN]->(o:Order
{number: $order_number}) RETURN SUM(i.cost)",
order_number=order_number)
        return result.values()

# find all Orders of a specific Customer
def find_orders_of_customer(customer_name):
    with driver.session() as session:
        result = session.run("MATCH (c:Customer {name:
$customer_name})-[:PLACED]->(o:Order) RETURN o",
customer_name=customer_name)
        return result.values()

# find all Items purchased by a specific Customer (via Order)
def find_items_purchased_by_customer(customer_name):
    with driver.session() as session:
        result = session.run("MATCH (c:Customer {name:
$customer_name})-[:PLACED]->(o:Order)<-[:INCLUDED_IN]-(i:Item) RETURN
i", customer_name=customer_name)
        return result.values()

# find the number of Items bought by a specific Customer (via Order)
def find_number_of_items_bought_by_customer(customer_name):
    with driver.session() as session:
        result = session.run("MATCH (c:Customer {name:
$customer_name})-[:PLACED]->(o:Order)<-[:INCLUDED_IN]-(i:Item) RETURN
COUNT(i)", customer_name=customer_name)
        return result.values()

# find for a Customer how much he bought items (via Order)
def find_total_cost_of_items_bought_by_customer(customer_name):
    with driver.session() as session:
        result = session.run("MATCH (c:Customer {name:
$customer_name})-[:PLACED]->(o:Order)<-[:INCLUDED_IN]-(i:Item) RETURN
```

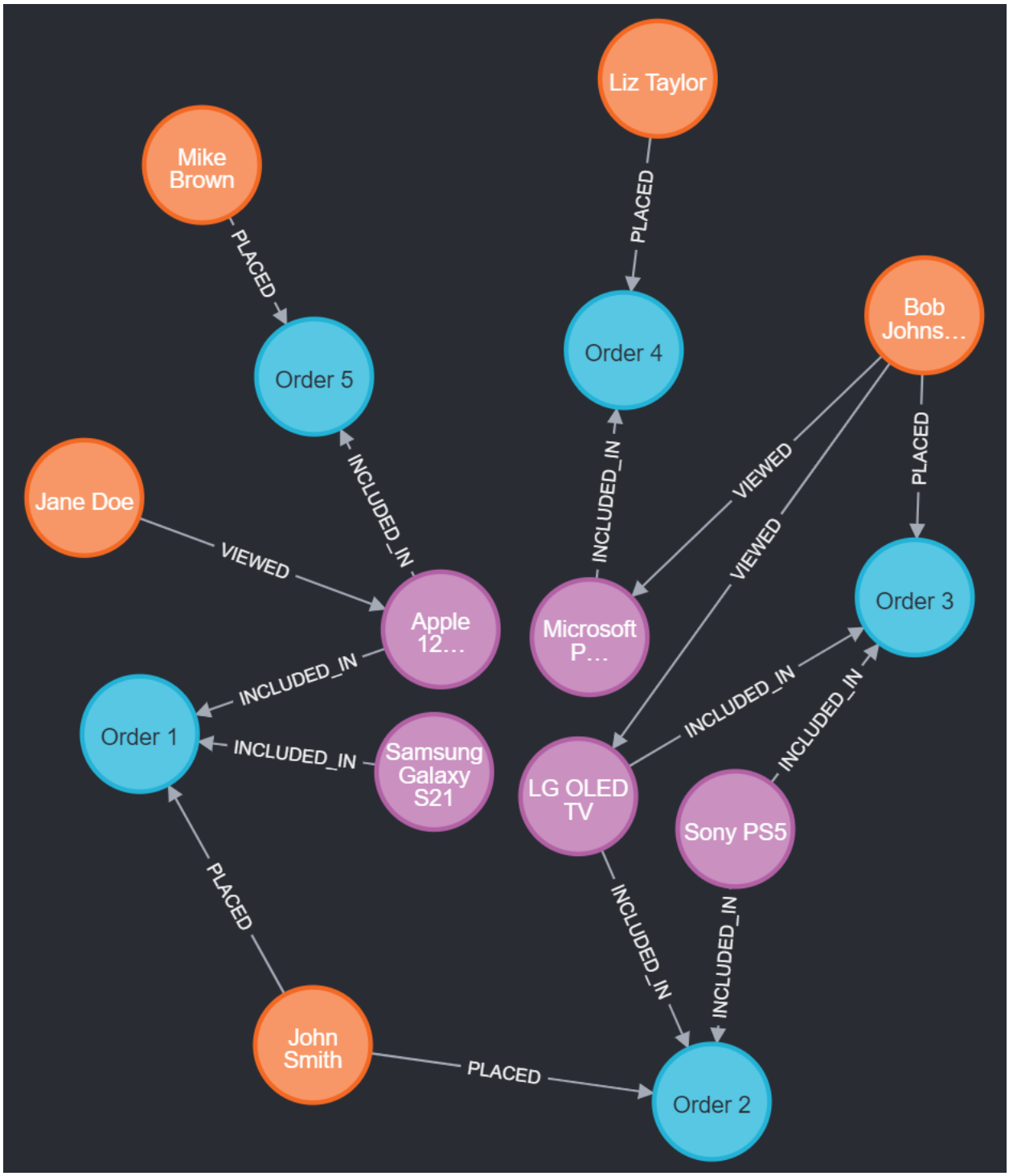
```
SUM(i.cost)", customer_name=customer_name)
        return result.values()

# find how many times each item was purchased, sort by this value
def find_number_of_times_each_item_was_purchased():
    with driver.session() as session:
        result = session.run("MATCH
(i:Item)-[:INCLUDED_IN]->(o:Order) RETURN i.name, COUNT(o) ORDER BY
COUNT(o) DESC")
        return result.values()

# find all Items viewed by a particular Customer
def find_items_viewed_by_customer(customer_name):
    with driver.session() as session:
        result = session.run("MATCH (c:Customer {name:
$customer_name})-[:VIEWED]->(i:Item) RETURN i",
customer_name=customer_name)
        return result.values()

# find other Items that were purchased together with a particular
Item (i.e. all Items that are included in the Order together with
this Item)
def find_items_purchased_together_with_item(item_name):
    with driver.session() as session:
        result = session.run("MATCH (i1:Item {name:
$item_name})-[:INCLUDED_IN]->(o:Order)<-[:INCLUDED_IN]-(i2:Item)
WHERE i1 <> i2 RETURN i2", item_name=item_name)
        return result.values()

# find Customers who bought this particular Item
def find_customers_who_bought_item(item_name):
    with driver.session() as session:
        result = session.run("MATCH (i:Item {name:
$item_name})-[:INCLUDED_IN]->(o:Order)<-[:PLACED]-(c:Customer) RETURN
c", item_name=item_name)
        return result.values()

# find items for a specific Customer that they viewed but did not buy
def find_items_viewed_but_not_bought_by_customer(customer_name):
    with driver.session() as session:
        result = session.run("MATCH (c:Customer {name:
$customer_name})-[:VIEWED]->(i:Item) WHERE NOT
(c)-[:PLACED]->(:Order)<-[:INCLUDED_IN]-(i) RETURN i",
```

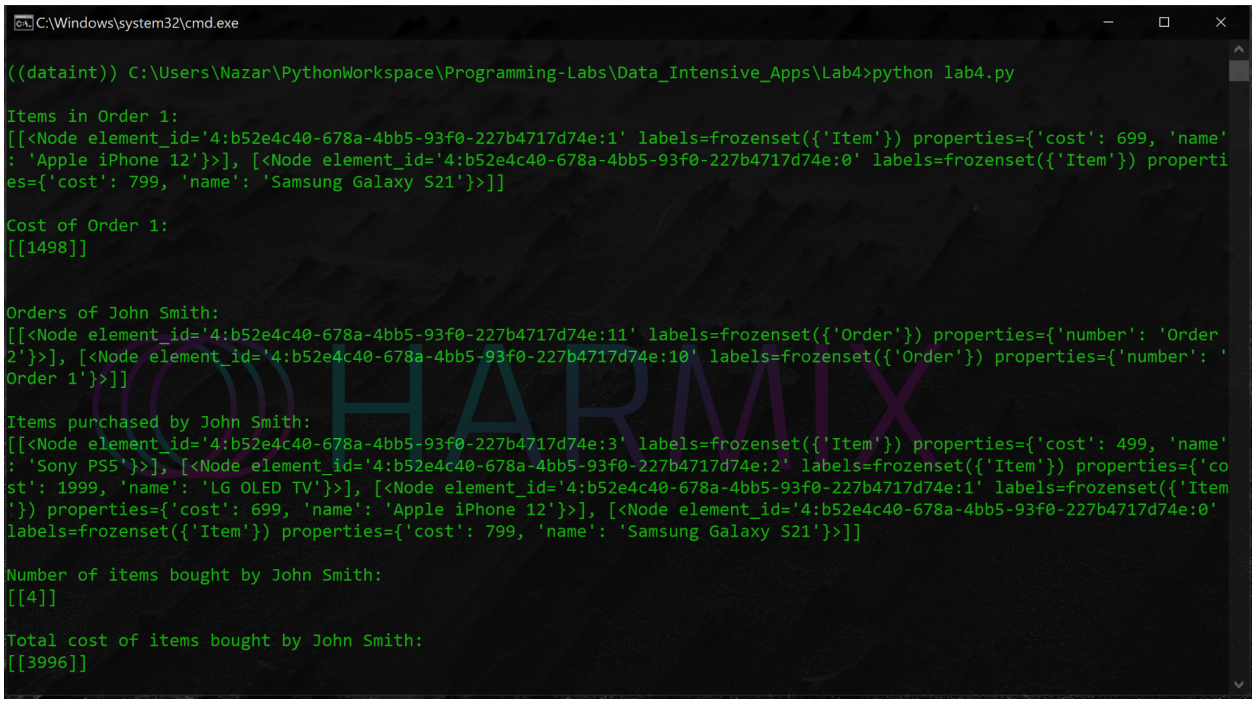
```
customer_name=customer_name)
        return result.values()


if __name__ == "__main__":
    # Docker: docker run -it --rm -p7474:7474 -p7687:7687 --name
testneo4j --env NEO4J_AUTH=neo4j/password neo4j
    driver = GraphDatabase.driver("bolt://localhost:7687",
auth=("neo4j", "password"))

    with driver.session() as session:
        session.run("MATCH (a)-[r]->() DELETE a, r")
        session.run("MATCH (a) DELETE a")

        session.run("CREATE (i1:Item {name: 'Samsung Galaxy S21',
cost: 799})")
        session.run("CREATE (i2:Item {name: 'Apple iPhone 12', cost:
699})")
        session.run("CREATE (i3:Item {name: 'LG OLED TV', cost:
1999})")
        session.run("CREATE (i4:Item {name: 'Sony PS5', cost: 499})")
        session.run("CREATE (i5:Item {name: 'Microsoft Surface Pro',
cost: 999})")

        session.run("CREATE (c1:Customer {name: 'John Smith'})")
        session.run("CREATE (c2:Customer {name: 'Jane Doe'})")
        session.run("CREATE (c3:Customer {name: 'Bob Johnson'})")
        session.run("CREATE (c4:Customer {name: 'Liz Taylor'})")
        session.run("CREATE (c5:Customer {name: 'Mike Brown'})")

        session.run("CREATE (o1:Order {number: 'Order 1'})")
        session.run("CREATE (o2:Order {number: 'Order 2'})")
        session.run("CREATE (o3:Order {number: 'Order 3'})")
        session.run("CREATE (o4:Order {number: 'Order 4'})")
        session.run("CREATE (o5:Order {number: 'Order 5'})")

        session.run("MATCH (i1:Item {name: 'Samsung Galaxy
S21'}),(o1:Order {number: 'Order 1'}) CREATE
(i1)-[:INCLUDED_IN]->(o1)")
        session.run("MATCH (i2:Item {name: 'Apple iPhone
12'}),(o1:Order {number: 'Order 1'}) CREATE
(i2)-[:INCLUDED_IN]->(o1)")
        session.run("MATCH (i3:Item {name: 'LG OLED TV'}),(o2:Order
```

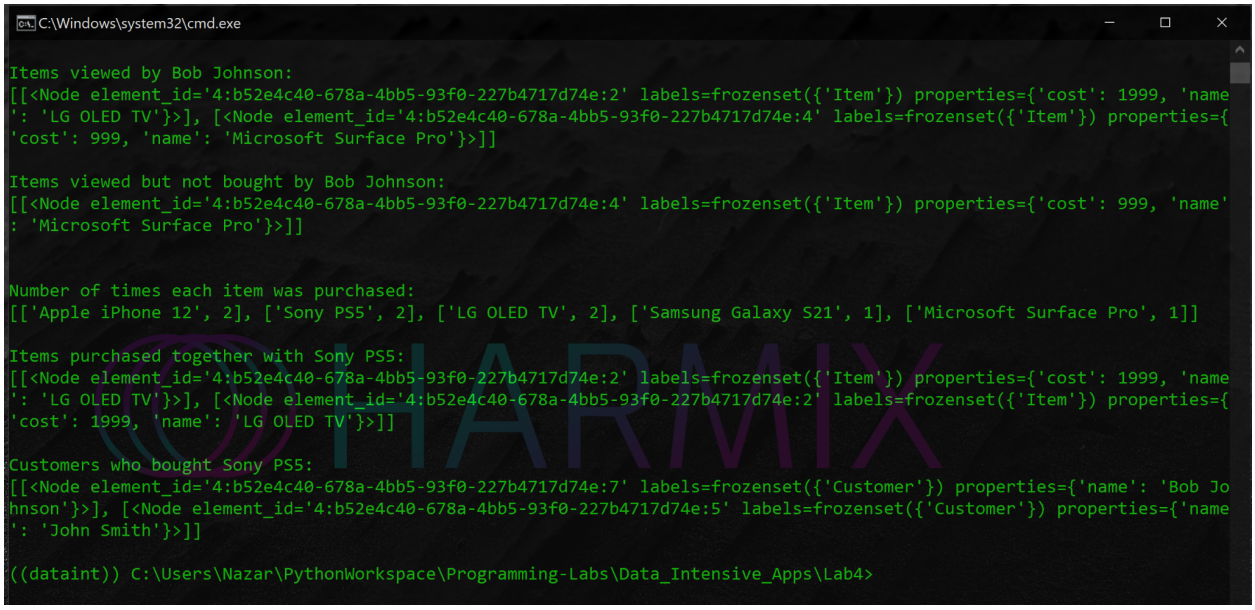
```
{number: 'Order 2'}) CREATE (i3)-[:INCLUDED_IN]->(o2)")
        session.run("MATCH (i4:Item {name: 'Sony PS5'}),(o2:Order
{number: 'Order 2'}) CREATE (i4)-[:INCLUDED_IN]->(o2)")
        session.run("MATCH (i3:Item {name: 'LG OLED TV'}),(o3:Order
{number: 'Order 3'}) CREATE (i3)-[:INCLUDED_IN]->(o3)")
        session.run("MATCH (i4:Item {name: 'Sony PS5'}),(o3:Order
{number: 'Order 3'}) CREATE (i4)-[:INCLUDED_IN]->(o3)")
        session.run("MATCH (i5:Item {name: 'Microsoft Surface
Pro'}),(o4:Order {number: 'Order 4'}) CREATE
(i5)-[:INCLUDED_IN]->(o4)")
        session.run("MATCH (i2:Item {name: 'Apple iPhone
12'}),(o5:Order {number: 'Order 5'}) CREATE
(i2)-[:INCLUDED_IN]->(o5)")

        session.run("MATCH (c1:Customer {name: 'John
Smith'}),(o1:Order {number: 'Order 1'}) CREATE (c1)-[:PLACED]->(o1)")
        session.run("MATCH (c1:Customer {name: 'John
Smith'}),(o2:Order {number: 'Order 2'}) CREATE (c1)-[:PLACED]->(o2)")
        session.run("MATCH (c3:Customer {name: 'Bob
Johnson'}),(o3:Order {number: 'Order 3'}) CREATE
(c3)-[:PLACED]->(o3)")
        session.run("MATCH (c4:Customer {name: 'Liz
Taylor'}),(o4:Order {number: 'Order 4'}) CREATE
(c4)-[:PLACED]->(o4)")
        session.run("MATCH (c5:Customer {name: 'Mike
Brown'}),(o5:Order {number: 'Order 5'}) CREATE (c5)-[:PLACED]->(o5)")

        session.run("MATCH (c2:Customer {name: 'Jane Doe'}),(i2:Item
{name: 'Apple iPhone 12'}) CREATE (c2)-[:VIEWED]->(i2)")
        session.run("MATCH (c3:Customer {name: 'Bob
Johnson'}),(i3:Item {name: 'LG OLED TV'}) CREATE
(c3)-[:VIEWED]->(i3)")
        session.run("MATCH (c3:Customer {name: 'Bob
Johnson'}),(i5:Item {name: 'Microsoft Surface Pro'}) CREATE
(c3)-[:VIEWED]->(i5)")

    order_number = "Order 1"
    print(f"\nItems in
{order_number}:\n{find_items_in_order(order_number)}")
    print(f"\nCost of
{order_number}:\n{calculate_cost_of_order(order_number)}")

    customer_name = "John Smith"
```

```
    print(f"\n\nOrders of
{customer_name}:\n{find_orders_of_customer(customer_name)}")
    print(f"\nItems purchased by
{customer_name}:\n{find_items_purchased_by_customer(customer_name)}")
    print(f"\nNumber of items bought by
{customer_name}:\n{find_number_of_items_bought_by_customer(customer_n
ame)}")
    print(f"\nTotal cost of items bought by
{customer_name}:\n{find_total_cost_of_items_bought_by_customer(custom
er_name)}")

    customer_name = "Bob Johnson"
    print(f"\n\nItems viewed by
{customer_name}:\n{find_items_viewed_by_customer(customer_name)}")
    print(f"\nItems viewed but not bought by
{customer_name}:\n{find_items_viewed_but_not_bought_by_customer(custo
mer_name)}")

    item_name = "Sony PS5"
    print(f"\n\nNumber of times each item was
purchased:\n{find_number_of_times_each_item_was_purchased()}")
    print(f"\nItems purchased together with
{item_name}:\n{find_items_purchased_together_with_item(item_name)}")
    print(f"\nCustomers who bought
{item_name}:\n{find_customers_who_bought_item(item_name)}")

    driver.close()
```

3) Знімки екрана з логами виконання:


Liz Taylor
Mike Brown
PLACED
PLACED
Order 4
Order 5
Bob Johns…
Jane Doe
INCLUDED_IN
INCLUDED_IN
VIEWED
VIEWED
VIEWED
PLACED
Order 3
Apple 12…
Microsoft P…
INCLUDED_IN
INCLUDED_IN
INCLUDED_IN
Order 1
INCLUDED_IN
Samsung Galaxy S21
LG OLED TV
Sony PS5
INCLUDED_IN
INCLUDED_IN
PLACED
John Smith
PLACED
Order 2


C:\Windows\system32\cmd.exe
((dataint)) C:\Users\Nazar\PythonWorkspace\Programming-Labs\Data_Intensive_Apps\Lab4>python lab4.py
Items in Order 1:
[[<Node element_id='4:b52e4c40-678a-4bb5-93f0-227b4717d74e:1' labels=frozenset({'Item'}) properties={'cost': 699, 'name': 'Apple iPhone 12'}>], [<Node element_id='4:b52e4c40-678a-4bb5-93f0-227b4717d74e:0' labels=frozenset({'Item'}) properties={'cost': 799, 'name': 'Samsung Galaxy S21'}>]]
Cost of Order 1:
[[1498]]
Orders of John Smith:
[[<Node element_id='4:b52e4c40-678a-4bb5-93f0-227b4717d74e:11' labels=frozenset({'Order'}) properties={'number': 'Order 2'}>], [<Node element_id='4:b52e4c40-678a-4bb5-93f0-227b4717d74e:10' labels=frozenset({'Order'}) properties={'number': 'Order 1'}>]]
Items purchased by John Smith:
[[<Node element_id='4:b52e4c40-678a-4bb5-93f0-227b4717d74e:3' labels=frozenset({'Item'}) properties={'cost': 499, 'name': 'Sony PS5'}>], [<Node element_id='4:b52e4c40-678a-4bb5-93f0-227b4717d74e:2' labels=frozenset({'Item'}) properties={'cost': 1999, 'name': 'LG OLED TV'}>], [<Node element_id='4:b52e4c40-678a-4bb5-93f0-227b4717d74e:1' labels=frozenset({'Item'}) properties={'cost': 699, 'name': 'Apple iPhone 12'}>], [<Node element_id='4:b52e4c40-678a-4bb5-93f0-227b4717d74e:0' labels=frozenset({'Item'}) properties={'cost': 799, 'name': 'Samsung Galaxy S21'}>]]
Number of items bought by John Smith:
[[4]]
Total cost of items bought by John Smith:
[[3996]]


C:\Windows\system32\cmd.exe
Items viewed by Bob Johnson:
[[<Node element_id='4:b52e4c40-678a-4bb5-93f0-227b4717d74e:2' labels=frozenset({'Item'}) properties={'cost': 1999, 'name': 'LG OLED TV'}>], [<Node element_id='4:b52e4c40-678a-4bb5-93f0-227b4717d74e:4' labels=frozenset({'Item'}) properties={'cost': 999, 'name': 'Microsoft Surface Pro'}>]]
Items viewed but not bought by Bob Johnson:
[[<Node element_id='4:b52e4c40-678a-4bb5-93f0-227b4717d74e:4' labels=frozenset({'Item'}) properties={'cost': 999, 'name': 'Microsoft Surface Pro'}>]]
Number of times each item was purchased:
[['Apple iPhone 12', 2], ['Sony PS5', 2], ['LG OLED TV', 2], ['Samsung Galaxy S21', 1], ['Microsoft Surface Pro', 1]]
Items purchased together with Sony PS5:
[[<Node element_id='4:b52e4c40-678a-4bb5-93f0-227b4717d74e:2' labels=frozenset({'Item'}) properties={'cost': 1999, 'name': 'LG OLED TV'}>], [<Node element_id='4:b52e4c40-678a-4bb5-93f0-227b4717d74e:2' labels=frozenset({'Item'}) properties={'cost': 1999, 'name': 'LG OLED TV'}>]]
Customers who bought Sony PS5:
[[<Node element_id='4:b52e4c40-678a-4bb5-93f0-227b4717d74e:7' labels=frozenset({'Customer'}) properties={'name': 'Bob Johnson'}>], [<Node element_id='4:b52e4c40-678a-4bb5-93f0-227b4717d74e:5' labels=frozenset({'Customer'}) properties={'name': 'John Smith'}>]]
((dataint)) C:\Users\Nazar\PythonWorkspace\Programming-Labs\Data_Intensive_Apps\Lab4>