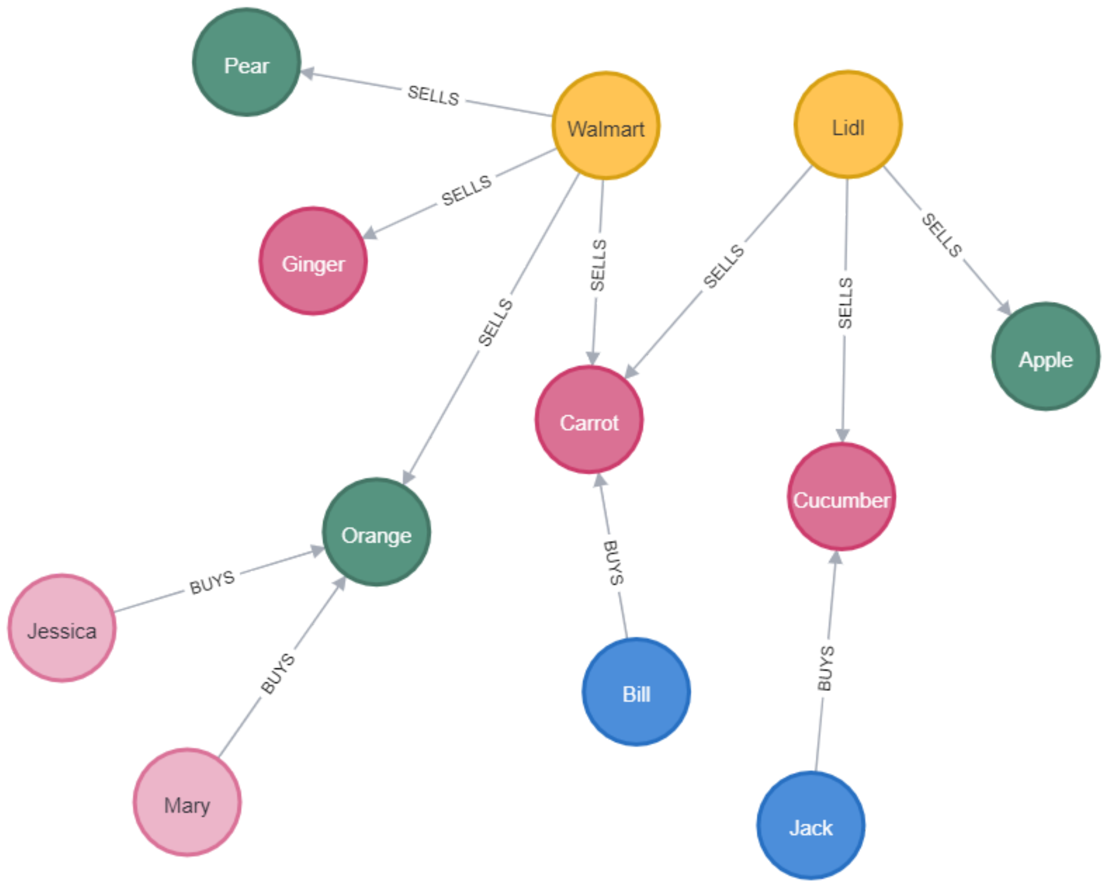
Graph Database Project



Our graph presents the relations between customers, buyers and the products.

Shop node stores the shop name and prices’ range in a given shop.

People are described by gender, date of birth and name.

Fruit and **Vegetable** node stores particular **product**’s name.

Sells relations have the price and quantity on stock attributes.

Buys relation describes purchase date, quantity bought and the shop where purchase was made.

Creates

```
CREATE (n:CUSTOMER:WOMAN {dateOfBirth:date("2019-09-10") ,name:'Jessica'});

CREATE (n:CUSTOMER:WOMAN {dateOfBirth:date("2000-09-10") ,name:'Mary'});

CREATE (n:CUSTOMER:MAN {dateOfBirth:date("2000-09-10") ,name:'Bill'});

CREATE (n:CUSTOMER:MAN {dateOfBirth:date("1998-09-10") ,name:'Jack'});

CREATE (n:SHOP {name:'Lidl', pricesRange:'medium'});

CREATE (n:SHOP {name:'Walmart', pricesRange:'low'});

CREATE (n:PRODUCT:FRUIT {name:'Orange'});

CREATE (n:PRODUCT:FRUIT {name:'Apple'});

CREATE (n:PRODUCT:FRUIT {name:'Pear'});
CREATE (n:PRODUCT:VEGATABLE {name:'Carrot'});

CREATE (n:PRODUCT:VEGATABLE {name:'Ginger'});

CREATE (n:PRODUCT:VEGATABLE {name:'Cucumber'});

MATCH (m:SHOP),(n:PRODUCT:VEGATABLE)
WHERE m.name = 'Lidl' AND n.name = 'Carrot'
  CREATE (m)-[:SELLS { price:7, quantity_on_stock:222}]->(n)
RETURN m,n;

MATCH (m:SHOP),(n:PRODUCT:VEGATABLE)
WHERE m.name = 'Lidl' AND n.name = 'Cucumber'
  CREATE (m)-[:SELLS { price:3, quantity_on_stock:1110}]->(n)
RETURN m,n;

MATCH (m:SHOP),(n:PRODUCT:VEGATABLE)
WHERE m.name = 'Walmart' AND n.name = 'Ginger'
  CREATE (m)-[:SELLS { price:33, quantity_on_stock:10}]->(n)
RETURN m,n;

MATCH (m:SHOP),(n:PRODUCT:VEGATABLE)
WHERE m.name = 'Walmart' AND n.name = 'Carrot'
  CREATE (m)-[:SELLS { price:11, quantity_on_stock:220}]->(n)
RETURN m,n;

MATCH (m:SHOP),(n:PRODUCT:FRUIT)
WHERE m.name = 'Walmart' AND n.name = 'Pear'
  CREATE (m)-[:SELLS { price:2, quantity_on_stock:20}]->(n)
RETURN m,n;

MATCH (m:SHOP),(n:PRODUCT:FRUIT)
WHERE m.name = 'Walmart' AND n.name = 'Orange'
  CREATE (m)-[:SELLS { price:6, quantity_on_stock:420}]->(n)
RETURN m,n;

MATCH (m:SHOP),(n:PRODUCT:FRUIT)
WHERE m.name = 'Lidl' AND n.name = 'Apple'
  CREATE (m)-[:SELLS { price:126, quantity_on_stock:440}]->(n)
```

```
RETURN m,n;

MATCH (m:CUSTOMER:WOMAN),(n:PRODUCT:FRUIT)
WHERE m.name = 'Jessica' AND n.name = 'Orange'
  CREATE (m)-[:BUYS {quantity_bought:4, purchase_date:date("2022-09-24"), shop_name:'Lidl'}]->(n)
RETURN m,n;

MATCH (m:CUSTOMER:WOMAN),(n:PRODUCT:FRUIT)
WHERE m.name = 'Mary' AND n.name = 'Orange'
  CREATE (m)-[:BUYS {quantity_bought:1, purchase_date:date("2022-09-24"), shop_name:'Lidl'}]->(n)
RETURN m,n;

MATCH (m:CUSTOMER:MAN),(n:PRODUCT:VEGATABLE)
WHERE m.name = 'Jack' AND n.name = 'Cucumber'
  CREATE (m)-[:BUYS {quantity_bought:33, purchase_date:date("2022-09-22"), shop_name:'Lidl'}]->(n)
RETURN m,n;

MATCH (m:CUSTOMER:MAN),(n:PRODUCT:VEGATABLE)
WHERE m.name = 'Bill' AND n.name = 'Carrot'
  CREATE (m)-[:BUYS {quantity_bought:2, purchase_date:date("2022-09-30"), shop_name:'Walmart'}]->(n)
RETURN m,n;
```

Competency questions

What is the average age of orange customers?

MATCH (c:CUSTOMER)-[:BUYS]->(p:PRODUCT)
WHERE p.name = "Orange"
RETURN avg(date().year - c.dateOfBirth.year)

	avg(date().year - c.dateOfBirth.year)
1	12.5

What is the most popular product among men?

MATCH (m:MAN)-[:BUYS]->(p:PRODUCT)
RETURN m.name, p.name, count(p)
ORDER BY count(p) DESC

	m.name	p.name	count(p)
1	"Bill"	"Carrot"	1
2	"Jack"	"Cucumber"	1

What is the product that Jack buys the most?

MATCH (m:MAN)-[:BUYS]->(p:PRODUCT)
WHERE m.name = "Jack"
RETURN m.name, p.name, count(p)
ORDER BY count(p) DESC LIMIT 1

m.name	p.name	count(p)
"Jack"	"Cucumber"	1

How many carrots were bought?

MATCH (c:CUSTOMER)-[b:BUYS]->(p:PRODUCT)
WHERE p.name = "Carrot" RETURN p.name, sum(b.quantity_bought)

p.name	sum(b.quantity_bought)
"Carrot"	2

Which shop sold more types of products?
MATCH (c:CUSTOMER)-[:BUYS]->(p:PRODUCT)<-[:SELLS]-(s:SHOP)
RETURN s.name, count(p), p.name

s.name	count(p)	p.name
"Walmart"	2	"Orange"
"Walmart"	1	"Carrot"
"Lidl"	1	"Carrot"
"Lidl"	1	"Cucumber"

Which shop has the highest average customer age?
MATCH (c:CUSTOMER)-[:BUYS]->(p:PRODUCT)<-[:SELLS]-(s:SHOP)
RETURN s.name, avg(date().year - c.dateOfBirth.year)
ORDER BY avg(date().year - c.dateOfBirth.year) DESC LIMIT 1

s.name	avg(date().year - c.dateOfBirth.year)
"Lidl"	23.0

Which shop has the most male customers?
MATCH (m:MAN:CUSTOMER)-[:BUYS]->(p:PRODUCT)<-[:SELLS]-(s:SHOP)
RETURN s.name, count(m)
ORDER BY count(m) DESC LIMIT 1

s.name	count(m)
"Lidl"	2

What is the amount of oranges sold by Lidl?
MATCH (c:CUSTOMER)-[b:BUYS]->(p:PRODUCT)
WHERE b.shop_name = "Lidl" AND p.name = "Orange"
RETURN b.shop_name, sum(b.quantity_bought)

b.shop_name	sum(b.quantity_bought)
"Lidl"	5

Which customer bought the biggest amount of products?

```
MATCH (c:CUSTOMER)-[b:BUYS]->(p:PRODUCT)
RETURN c.name, sum(b.quantity_bought)
ORDER BY sum(b.quantity_bought) DESC LIMIT 1
```

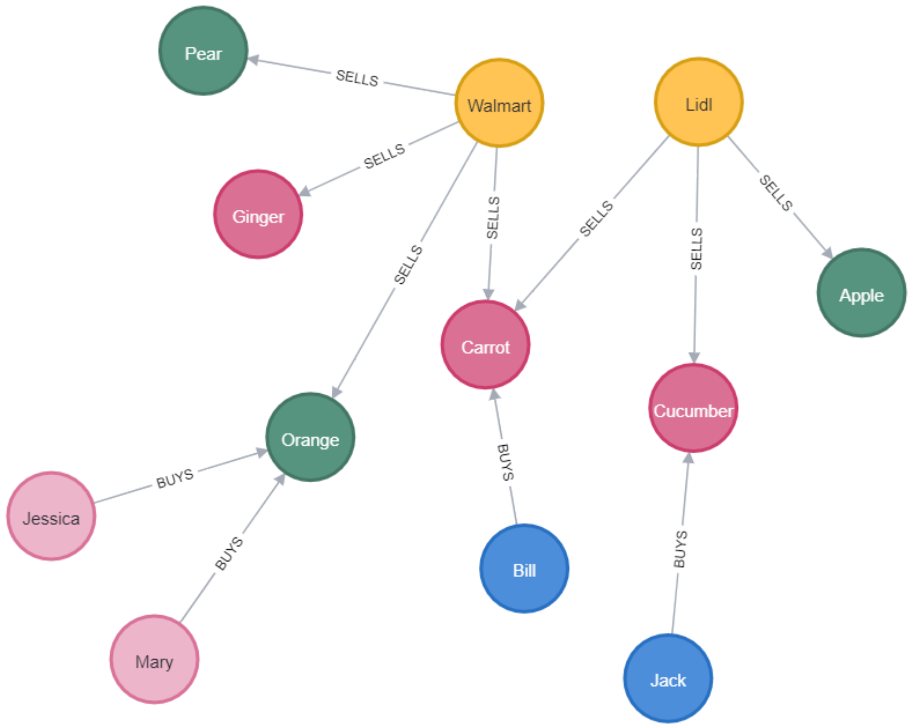
c.name	sum(b.quantity_bought)
"Jack"	33

Who bought the biggest amount of vegetables?

```
MATCH (c:CUSTOMER)-[b:BUYS]->(p:PRODUCT:VEGATABLE)
RETURN c.name, sum(b.quantity_bought)
ORDER BY sum(b.quantity_bought) DESC LIMIT 1
```

c.name	sum(b.quantity_bought)
"Jack"	33

Graph Analyzis



Degree centrality - shows how many buyers each fruit has

- **Degree centrality** measures the number of incoming and outgoing relationships from a node. The Degree Centrality algorithm can help us find popular nodes in a graph.

MATCH (u:PRODUCT:FRUIT)
RETURN u.name AS name,
size((u)<-[:BUYS]-()) AS buyers

	name	buyers
1	"Orange"	2
2	"Apple"	0
3	"Pear"	0

The Triangle Count algorithm counts the number of triangles for each node in the graph.

```
MATCH (n)-[]->(z)-[]->(n)
RETURN count(n)
```

	count(n)
1	0

Jaccard Similarity - measures the similarity between two sets of data to see which members are shared and distinct. It is defined as the size of the intersection divided by the size of the union of two sets.

```
MATCH (p1:SHOP {name: 'Walmart'})-[:SELLS]->(s1)
WITH p1, collect(id(s1)) AS p1s
MATCH (p2:SHOP)-[:SELLS]->(s2) WHERE p1 <> p2
WITH p1, p1s, p2, collect(id(s2)) AS p2s
RETURN p1.name AS from, p2.name AS to,
  gds.similarity.jaccard(p1s, p2s) AS jaccard;
```

	from	to	jaccard
1	"Walmart"	"Lidl"	0.16666666666666666