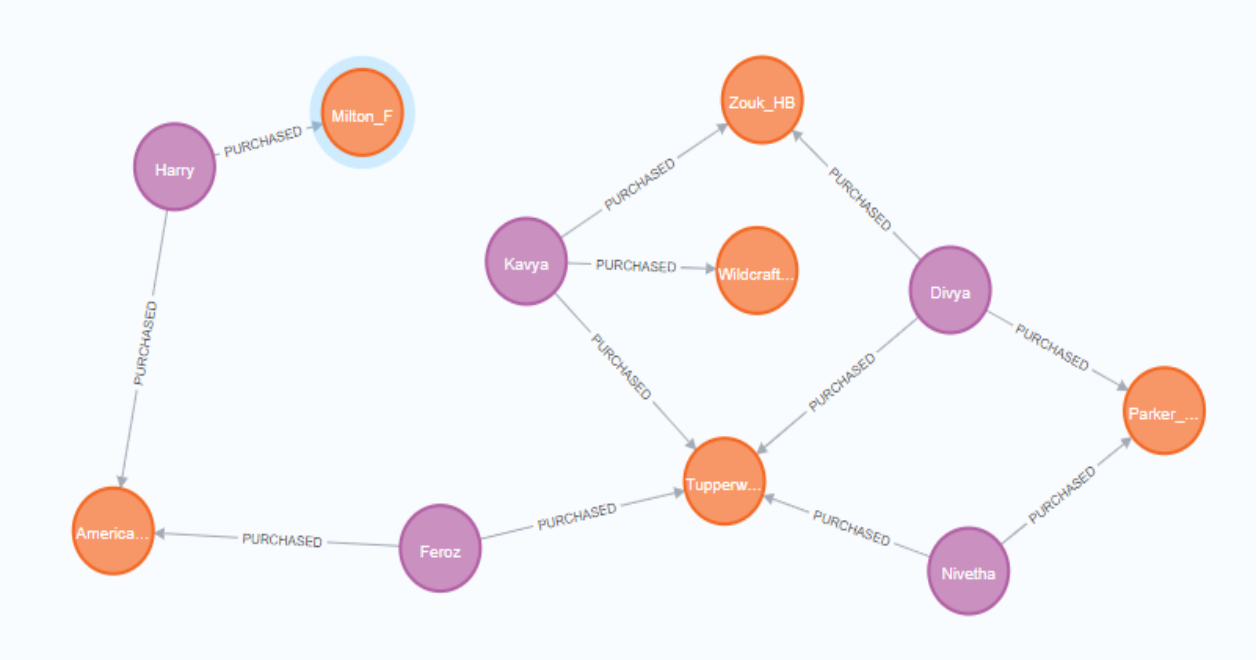
GRAPH:



Having Buyer node and Product nodes that are connected by a relationship called Purchased.

SORTING:

- → **In data structure, the arrangement of data in a preferred ordered is called Sorting.**
- → Sorting is **the processing of arranging the data in ascending and descending order**.
- → By sorting data, it is easier to search through it quickly and easily.

Sorted the Buyer list by their name,

MATCH (B:buyer)

RETURN B.name,B.contactno,B.email,B.address,B.gender

ORDER BY B.name

```
MATCH (B:buyer)
RETURN B.name,B.contactno,B.email,B.address,B.gender
ORDER BY B.name
```

	B.name	B.contactno	B.email	B.address	B.gender
1	"Divya"	9789900998	"dd@gmail.com"	"chennai"	"female"
2	"Feroz"	7010609382	"fero@gmail.com"	"chennai"	"male"
3	"Harry"	9232425262	"hp@gmail.com"	"salem"	"male"
4	"Kavya"	9789470466	"che@gmail.com"	"chennai"	"female"
5	"Nivetha"	9629862285	"nivi@gmail.com"	"chennai"	"female"

Sorted the Product list by the Product_id,

MATCH (P:product)

RETURN P.P_id,P.P_name,P.P_rate,P.P_stock

ORDER BY P.P_id

```
neo4j$ MATCH (P:product) RETURN P.P_id,P.P_name,P.P_rate,P.P_stock ORDER BY P.P_id
```

	P.P_id	P.P_name	P.P_rate	P.P_stock
1	1	"Wildcraft_TB"	1000	"YES"
2	2	"Milton_F"	600	"YES"
3	3	"Parker_FP"	250	"YES"
4	4	"Zouk_HB"	2000	"YES"
5	5	"Tupperware_B"	1000	"YES"
6	6	"AmericanTourister_BP"	1300	"YES"

SEARCHING:

→ Searching in data-structure refers to the process of finding a desired element in a set of items.
→ The desired element is called "target".
→ Search refers to locating a desired element of specified properties in a collection of items.

Searched the Products that are all in stock,

MATCH (P:product) WHERE P.P_stock="YES" RETURN P.P_id,P.P_name,P.P_rate,P.P_stock

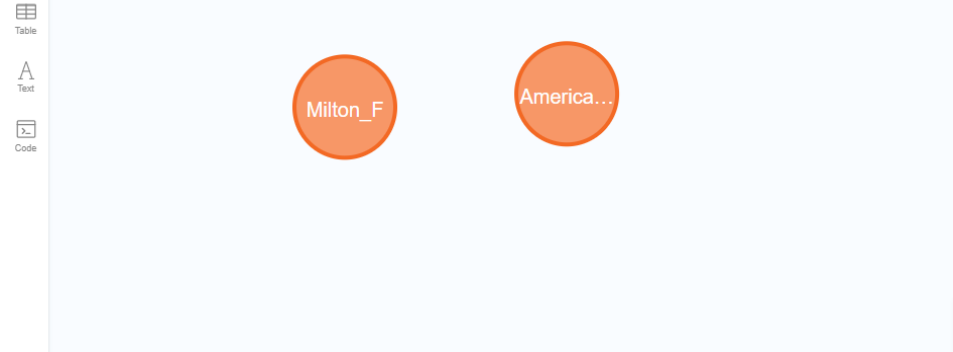
```
neo4j$ MATCH (P:product) WHERE  P.P_stock="YES" RETURN P.P_id,P.P_name,P.P_rate,P.P_stock
```

	P.P_id	P.P_name	P.P_rate	P.P_stock
1	1	"Wildcraft_TB"	1000	"YES"
2	3	"Parker_FP"	250	"YES"
3	4	"Zouk_HB"	2000	"YES"
4	5	"Tupperware_B"	1000	"YES"

Searched the Products that are NOT in stock,

MATCH (P:product) WHERE P.P_stock="NO" RETURN P

```
neo4j$ MATCH (P:product) WHERE P.P_stock="NO" RETURN P
```



MATCH (P:product) WHERE P.P_stock<>"YES" RETURN P.P_id,P.P_name,P.P_rate,P.P_stock

```
neo4j$ MATCH (P:product) WHERE P.P_stock◇"YES" RETURN P.P_id,P.P_name,P.P_rate,P.P_stock
```

	P.P_id	P.P_name	P.P_rate	P.P_stock
1	2	"Milton_F"	600	"NO"
2	6	"AmericanTourister_BP"	1300	"NO"

PAGINATION:

In Neo4j, paginating by using **Limit and Skip** clause.

Query using **Limit** clause,

MATCH (B:buyer)-[:PURCHASED]->(P:product)

RETURN B.name

LIMIT 3

```
neo4j$
neo4j$ MATCH (B:buyer)-[:PURCHASED]→(P:product) RETURN B.name LIMIT 3
```

	B.name
1	"Kavya"
2	"Harry"
3	"Nivetha"

Started streaming 3 records after 1 ms and completed after 8 ms.

Query using **Skip** clause,

MATCH(B:buyer) RETURN B.name ORDER BY B.name SKIP 2

```
MATCH(B:buyer)
RETURN B.name
ORDER BY B.name
SKIP 2
```

	B.name
1	"Harry"
2	"Kavya"
3	"Nivetha"

Fetching data by condition:

MATCH (B:buyer)-[:PURCHASED]->(P:product)

WHERE B.name="Kavya"

RETURN B.name as BUYER_NAME, P.P_name AS PRODUCT ,P.P_rate AS AMOUNT

```
neo4j$ MATCH (B:buyer)-[:PURCHASED]→(P:product) WHERE B.name="Kavya" RETURN B.name as BUYER_NAME, P.P_name A…
```

	BUYER_NAME	PRODUCT	AMOUNT
1	"Kavya"	"Zouk_HB"	2000
2	"Kavya"	"Tupperware_B"	1000
3	"Kavya"	"Wildcraft_TB"	1000

MATCH (B:buyer)-[:PURCHASED]->(P:product)

WHERE B.name="Kavya"

RETURN B.name AS BUYER_NAME,SUM(P.P_rate) AS TOTAL_BILL_AMOUNT

```
MATCH (B:buyer)-[:PURCHASED]→(P:product)
WHERE B.name="Kavya"
RETURN B.name AS BUYER_NAME,SUM(P.P_rate) AS TOTAL_BILL_AMOUNT
```

Table

Text

Code

	BUYER_NAME	TOTAL_BILL_AMOUNT
1	"Kavya"	4000