Final Project Part 3

Abhinav Kumar

Summary statistics and schema:

The dataset is of laptop pricing having different types of laptops along with different features like Memory, Operating System, etc. After loading the data in the Neo4j browser I got **10424 nodes** as shown in Figure 1 and **9121 relationships** as shown in Figure 2. Figure 3 demonstrates the schema diagram having **7 nodes** which are Product, Processing, Company, Memory Configuration, Price Euro, Operating System, Size Configuration, Type and Name. Also, it has **4 types** of **relationships** which are OF, TYPE, WITH and HAS. Also, the arrow diagram is added to get a clear picture of the nodes and their relationship as shown in Figure 4.

Exploratory data description and queries:

We will see some important ways how to get the best laptops from our data set. Money comes first into the mind while deciding on any kind of laptop a customer wants to buy.

How to get the average price of the laptop that is available in the dataset? For that, we are using the Price Euro Configuration node. It has price property having the price in Euro. Since the data is in the string it is converted into an integer by using **toInteger**. Taking the average price of the laptop by using the **avg()** and by using the query shown in Figure 5 we got the average size of the laptop is **14.5 inches**. Now RAM (Read Access Memory) and SSD (Solid State Drive) are also important aspects of a laptop. A good RAM of a laptop is 8GB RAM.

How to find all SSD are available for 8GB RAM? It can help the customers to easily select their correct laptop having a good laptop. Now by executing the query shown in Figure 6 we can find that **25 different types of SSD** available to choose from. Next part we will focus on how a laptop looks like means on dimension-wise.

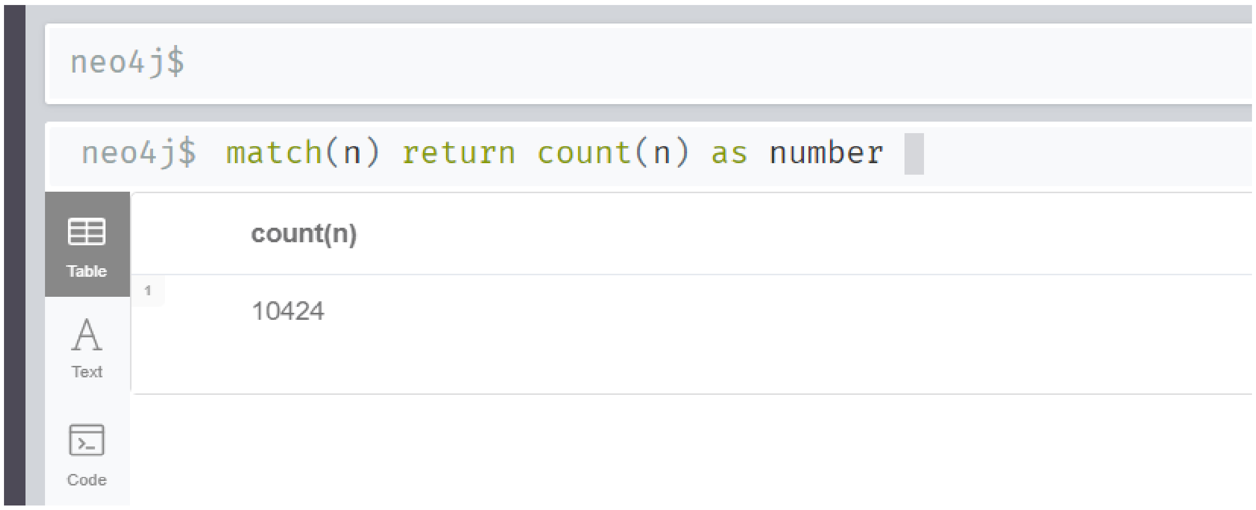
How to categorize the length of the laptop and get their respective weight and screen resolution? The length of the laptop is categorized as follows by using **apoc.case** query. The laptop size which is less than or equal to 13 inches comes under a small size. If the laptop size is greater than 13 inches and less than or equal to 15 inches then it will come under medium size. Furthermore, if the size of the laptop is greater than 15 inches then it will come under the larger size. Along with the size, the weight and screen resolution of the laptop is also being fetched. In figure 7, 17.3 inches was passed as a parameter and the result shows the laptop comes under the larger size category with different weight and resolution of the laptop in the same size. Since the length of the laptop are in float value so **toFloat** is used to convert string to float value before comparing**.** For all the comparisons the **greater than (>), greater than (>=), less than(<) and less than (<=)**operators are used. Lastly, the processor of the laptop which is the heart of any laptop has the greatest impact on productivity.

How to categorize the laptop based on the good processor? For this**,** I am fetching all the products in which different types of Intel processor is used. In Figure 8, the relationship of the product with the processing unit and two types of intel processors are checked. For this, **IN()** operator is used. To get the distinct output **DISTINCT** is used. For convenience, the list of products that is the output is **ORDER BY DESC** which helps to easily visualize the result. Finally, 129 laptop products are available for the Intel processor as demonstrated in Figure 8.

REFERENCES

1. https://neo4j.com/docs/cypher-manual/current/functions/aggregating/
2. https://neo4j.com/developer/kb/conditional-cypher-execution/https:/neo4j.com/developer/kb/conditional-cypher-execution/
3. https://neo4j.com/docs/cypher-manual/current/syntax/operators/

APPENDIX

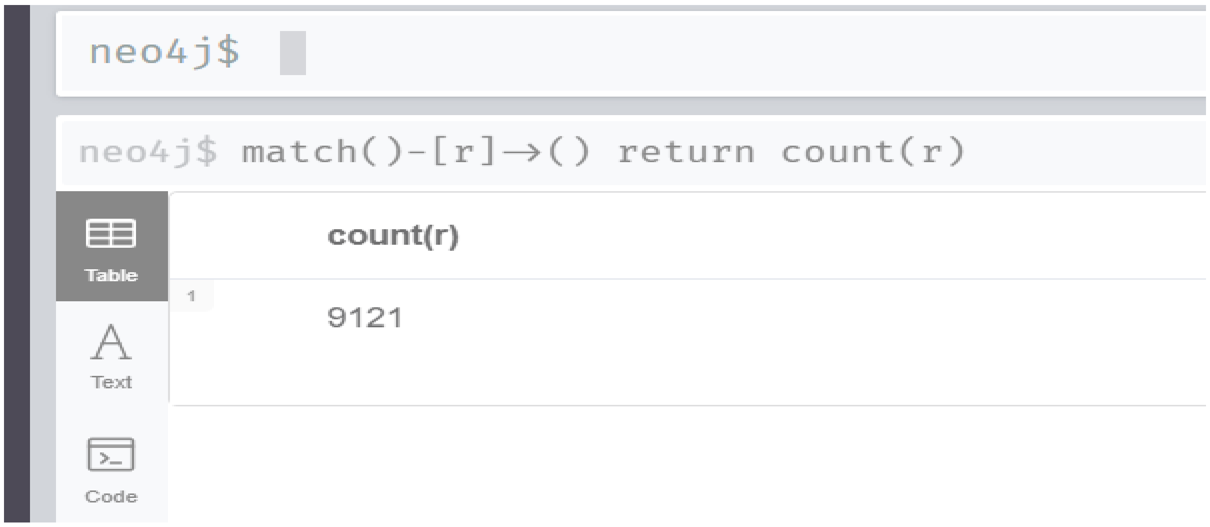


```
neo4j$

neo4j$ match(n) return count(n) as number

count(n)
1  10424
```

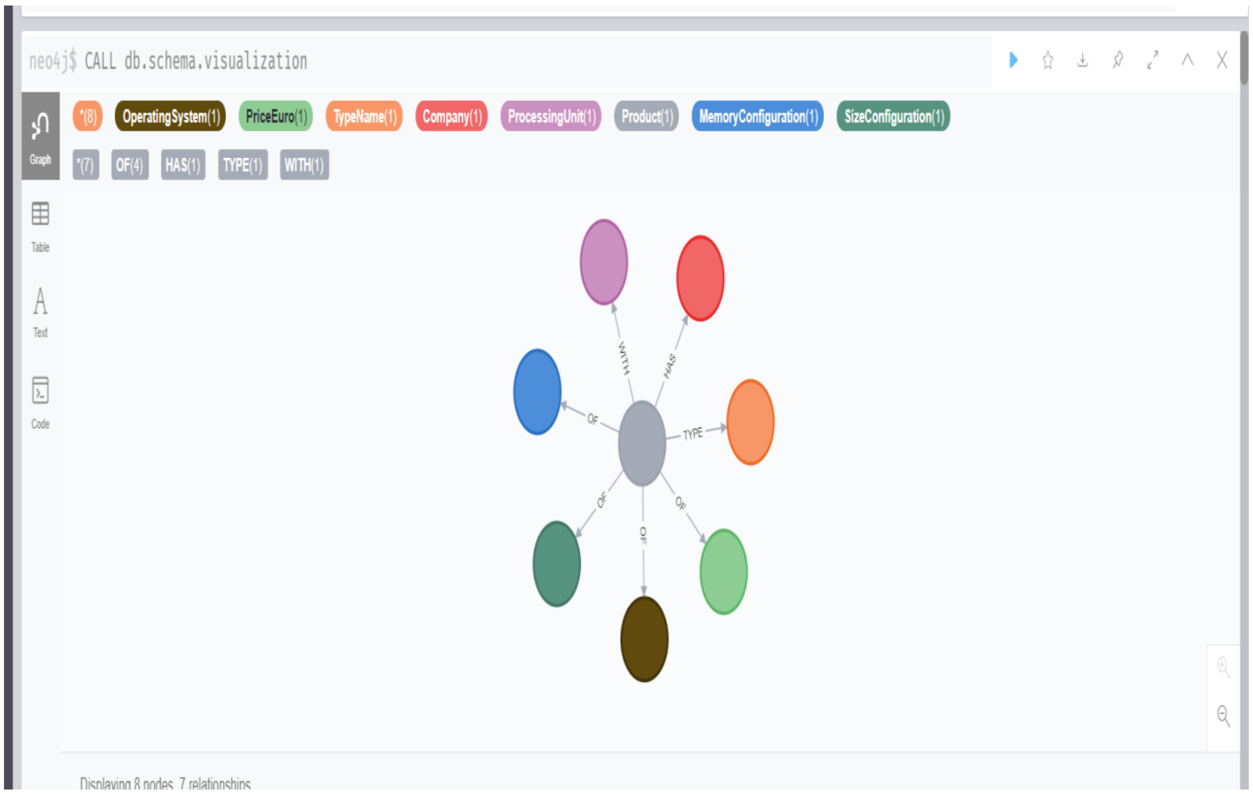
Figure 1. Nodes


```
neo4j$
```

```
neo4j$ match()-[r]→() return count(r)
```

```
count(r)
1  9121
```

Figure 2. Relationships



```
neo4j$ CALL db.schema.visualization
```

Displaying 8 nodes, 7 relationships

Figure 3. Schema

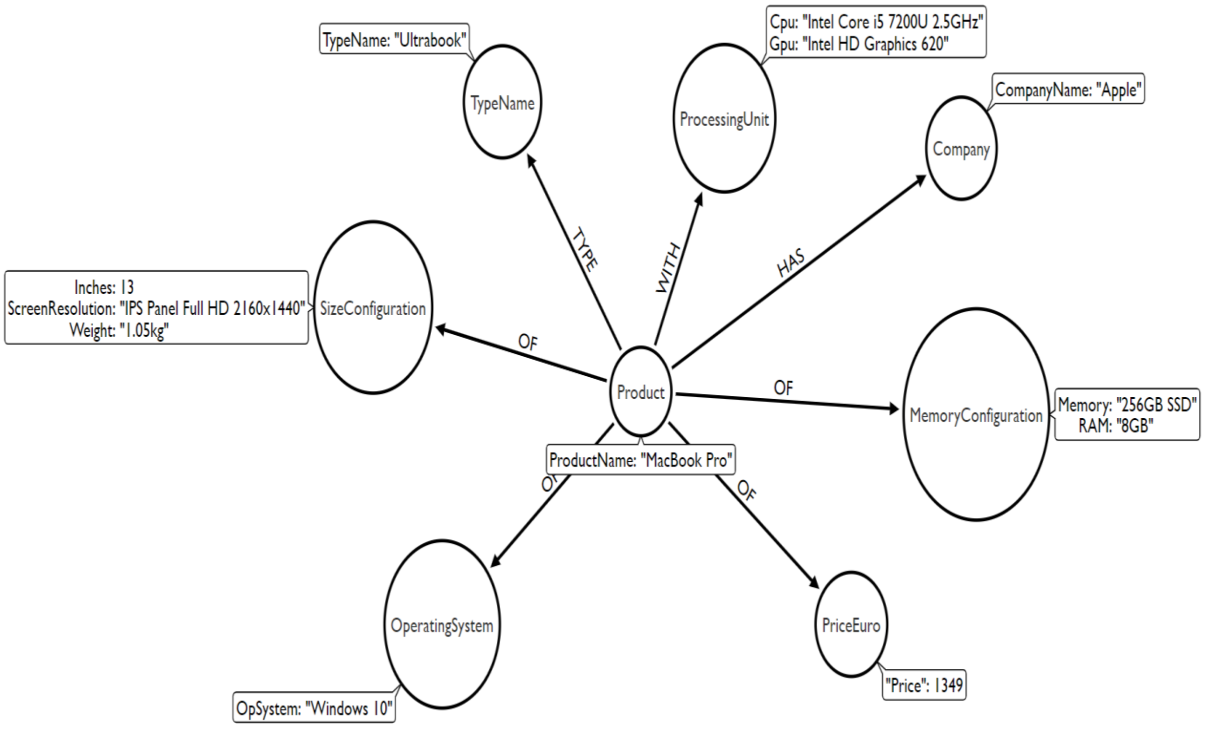
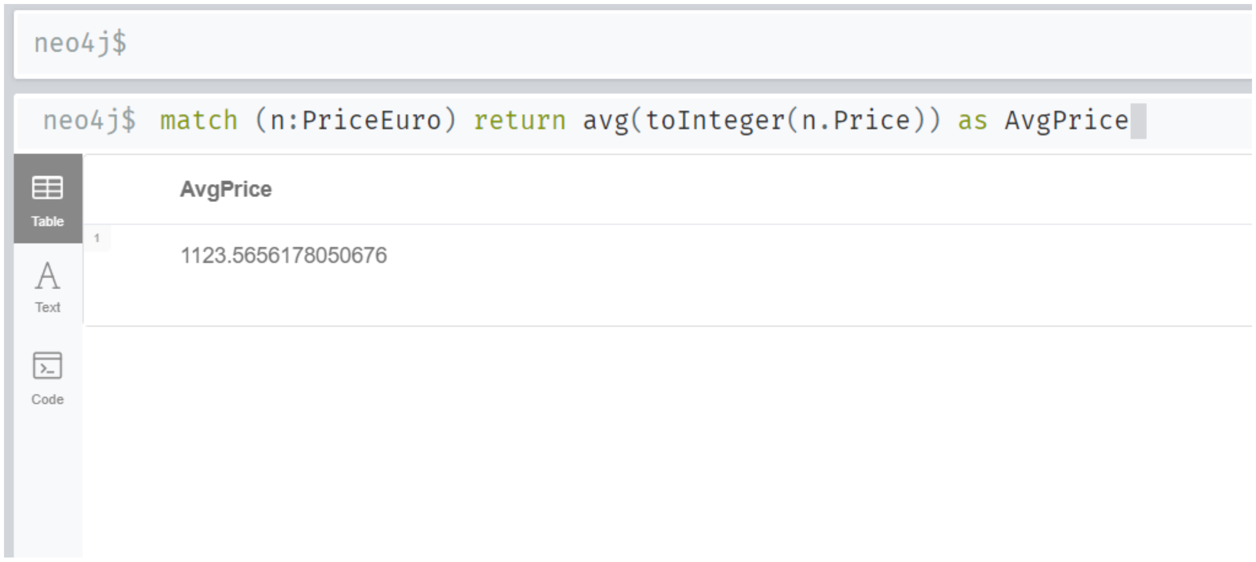


Figure 4. Arrow Diagram

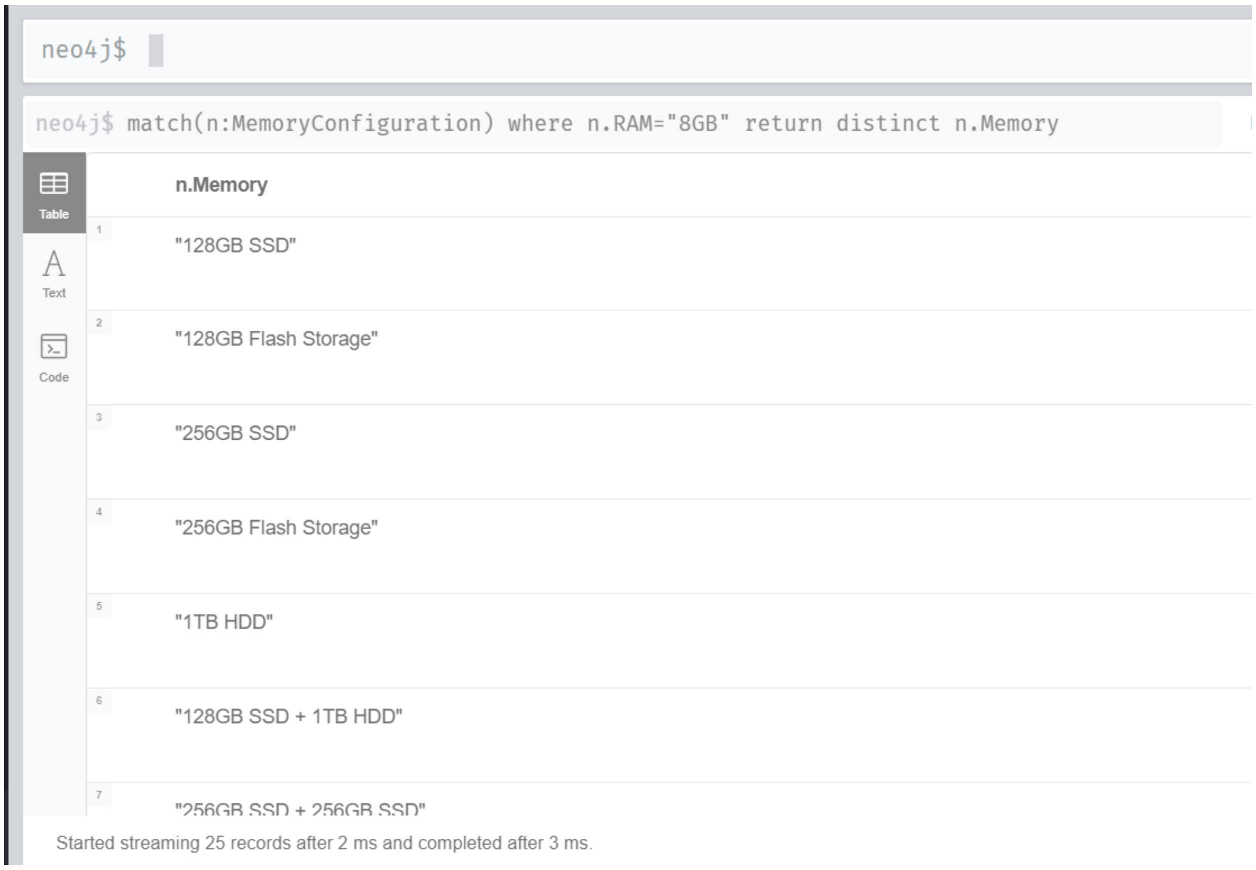


```
neo4j$

neo4j$ match (n:PriceEuro) return avg(toInteger(n.Price)) as AvgPrice
```

	AvgPrice
1	1123.5656178050676

Figure 5. Average Laptop Price



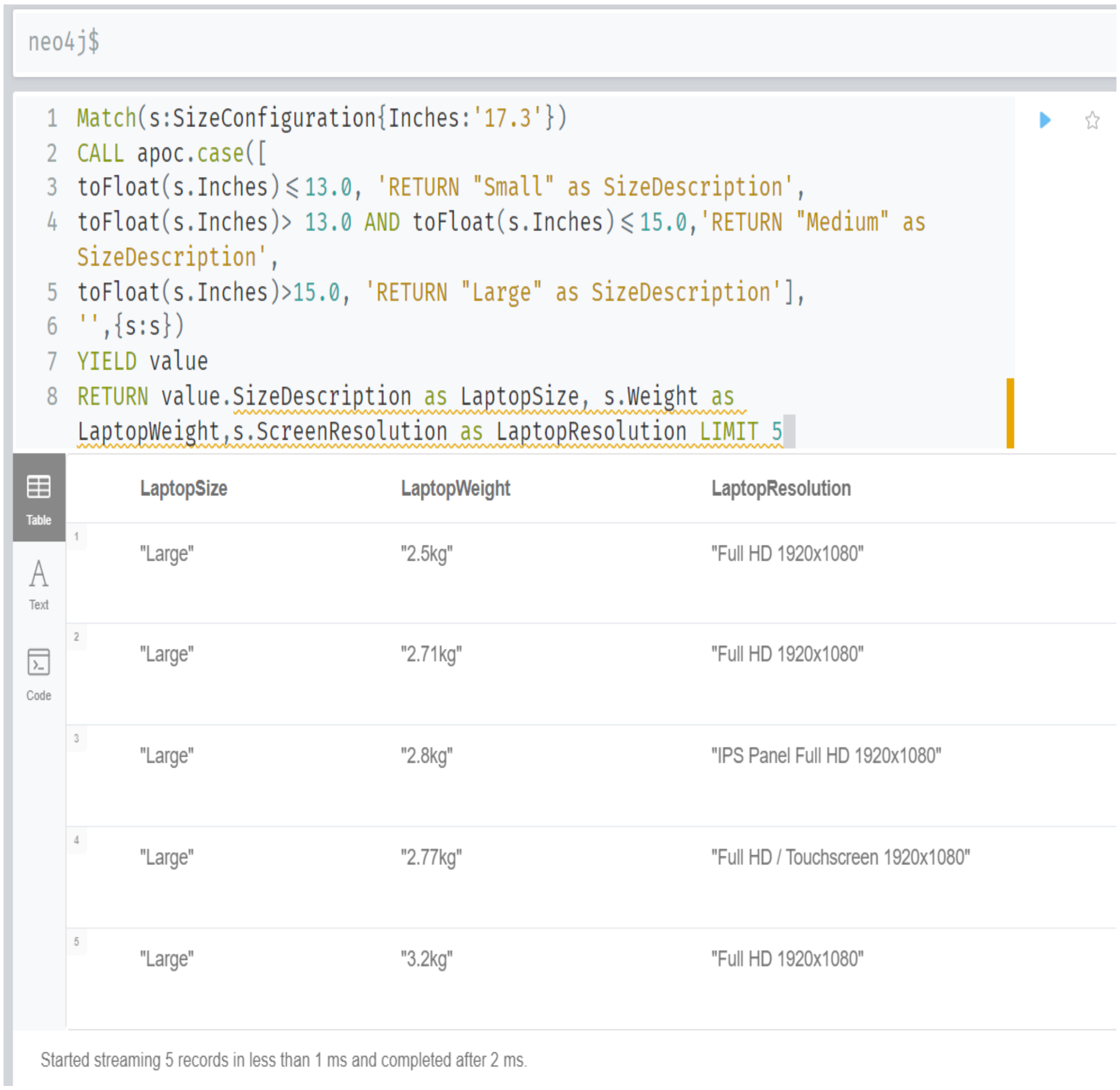
```
neo4j$

neo4j$ match(n:MemoryConfiguration) where n.RAM="8GB" return distinct n.Memory
```

```
   n.Memory
1  "128GB SSD"
2  "128GB Flash Storage"
3  "256GB SSD"
4  "256GB Flash Storage"
5  "1TB HDD"
6  "128GB SSD + 1TB HDD"
7  "256GB SSD + 256GB SSD"

Started streaming 25 records after 2 ms and completed after 3 ms.
```

Figure 6. Available Memories for 8GB RAM



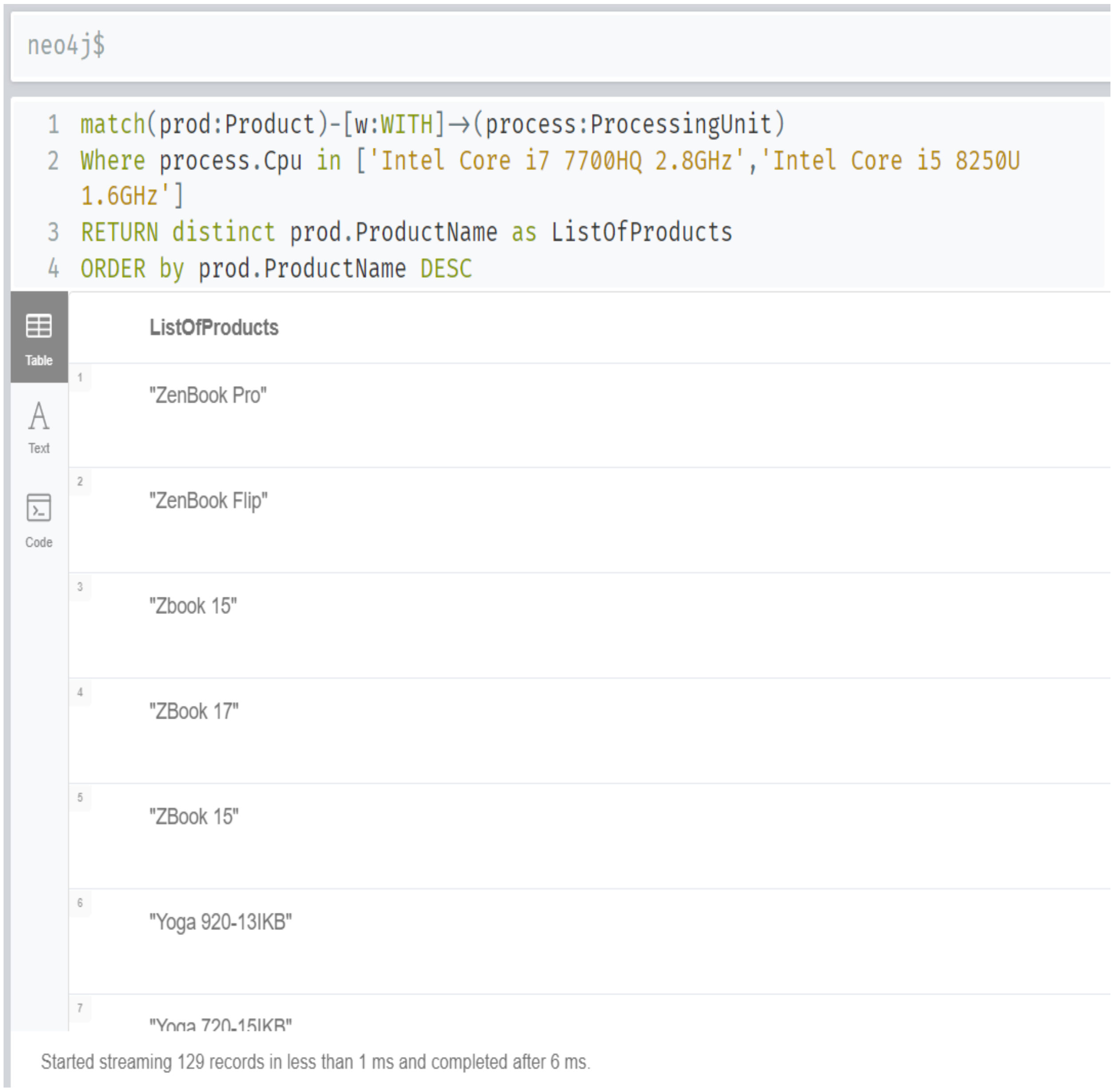
```
neo4j$
```

```
Match(s:SizeConfiguration{Inches:'17.3'})
CALL apoc.case([
toFloat(s.Inches)≤13.0, 'RETURN "Small" as SizeDescription',
toFloat(s.Inches)> 13.0 AND toFloat(s.Inches)≤15.0,'RETURN "Medium" as
SizeDescription',
toFloat(s.Inches)>15.0, 'RETURN "Large" as SizeDescription'],
'',{s:s})
YIELD value
RETURN value.SizeDescription as LaptopSize, s.Weight as
LaptopWeight,s.ScreenResolution as LaptopResolution LIMIT 5
```

	LaptopSize	LaptopWeight	LaptopResolution
1	"Large"	"2.5kg"	"Full HD 1920x1080"
2	"Large"	"2.71kg"	"Full HD 1920x1080"
3	"Large"	"2.8kg"	"IPS Panel Full HD 1920x1080"
4	"Large"	"2.77kg"	"Full HD / Touchscreen 1920x1080"
5	"Large"	"3.2kg"	"Full HD 1920x1080"

Started streaming 5 records in less than 1 ms and completed after 2 ms.

Figure 7. Categorizing Size, Weight & Resolution



```
neo4j$
```

```
match(prod:Product)-[w:WITH]→(process:ProcessingUnit)
Where process.Cpu in ['Intel Core i7 7700HQ 2.8GHz','Intel Core i5 8250U
1.6GHz']
RETURN distinct prod.ProductName as ListOfProducts
ORDER by prod.ProductName DESC
```

	ListOfProducts
1	"ZenBook Pro"
2	"ZenBook Flip"
3	"Zbook 15"
4	"ZBook 17"
5	"ZBook 15"
6	"Yoga 920-13IKB"
7	"Yoga 720-15IKB"

Started streaming 129 records in less than 1 ms and completed after 6 ms.

Figure 8. Products having Intel Core Processor