**Dataset used:** https://github.com/sonalt9/graph\_query\_performance/blob/main/e-commerce%20dataset.zip

**Code for Data Loading into Baseline Graph Model using Neo4j**

**// Customer Nodes**

LOAD CSV WITH HEADERS FROM "file:///Customer.csv"

AS line

CREATE (c:Customer {c\_id:toint(line.c\_id),con:line.con})

**// Order Nodes**

LOAD CSV WITH HEADERS FROM "file:///Order.csv" AS row

Create (o:Order {o\_id:toint(row.o\_id), o\_date: apoc.date.parse(row.o\_date, ‘DD-MM-YYYY’)})

**// hasOrder Relationship**

LOAD CSV WITH HEADERS FROM "file:///Order.csv" AS row

MATCH (o:Order {o\_id:toint(row.o\_id)})

MATCH (c:Customer {c\_id: toint(row.c\_id)})

MERGE (c)-[:hasOrder]->(o);

**// Product Nodes**

LOAD CSV WITH HEADERS FROM "file:///Product.csv" AS row

Create (p:Product {p\_id:toint(row.p\_id), p\_model:line.p\_model, p\_price: toint(line. p\_price), p\_br: line.p\_br, p\_os:line.p\_os, p\_pr:line.p\_processor})

**// hasProduct Relationship**

using periodic commit 1000

LOAD CSV WITH HEADERS FROM "file:///Order\_Details.csv" AS row

MATCH (o:Order {o\_id:toint(row.o\_id)})

MATCH (p:Product {p\_id: toint(row.p\_id)})

MERGE (o)-[:hasProduct]->(p);

**Code for Query Execution of Baseline Graph Model using Neo4j**

**Q1:** match(c:Customer{con:"France"}) return c

**Q2:** match (p:Product) return distinct p.p\_br

**Q3:** match (p:Product) return p.p\_br, count (p)

**Q4:** match(c:Customer)-[:hasOrder]->(o:Order)-[:hasProduct]->(p:Product) return p

**Q5:** match (c:Customer) return c order by c.con

**Q6:** match (n: Customer{con: 'Iceland'})-[:hasOrder]->(o:Order)-[:hasProduct]->(p:Product) return p

**Q7** match (p: Product{p\_id: 1})<-[:hasProduct]-(o:Order)<-[:hasOrder]-(c:Customer)

return c.con, count(c.con) order by count(c.con) desc limit 1

**Q8** match (p:Product{p\_id:10}) match (q: Product{p\_br:p.p\_br}) return q

**Q9** match(p:Product{p\_br:"Lava"})<-[:hasProduct]-(o:Order) return p, count(o) order by count(o) desc limit 1

**Q10** match(p:Product{p\_id:519})<-[:hasProduct]-(q:Order)-[:hasProduct]->(r:Product) return count(r), r order by count(r) desc limit 5