## 1. Looking at the Orders table, there’s a field called ShipStateProvince. Write a query that shows the OrderID, CustomerID, and ShipStateProvince for the orders where the ShipStateProvince is either France or Belgium.

### Postgres

```
 SELECT order_id, customer_id, ship_country
FROM orders
WHERE ship_country IN ('France', 'Belgium');
```

### MongoDB

```
 db.orders.find({ ship_state_province: { $in: [ "FL", "IL" ] } }, { _id: 1,  customer_id: 1, ship_state_province: 1 } )
```

### Neo4J

```

```

## 2. How many customers do we have in the Customers table? Show one

value only, and don’t rely on getting the recordcount at the end of a resultset.

### Postgres

```
 SELECT COUNT(*) AS TotalCustomers
FROM Customers;
```

### MongoDB

```
 db.customers.find( { "_id": { "$exists": true } } ).count()
```

### Neo4J

```
 MATCH (c:Customer)
RETURN count(c) AS TotalCustomers;
```

## 3. We’d like to show, for each product, the associated Supplier. Show the

ProductID, ProductName, and the CompanyName of the Supplier. Sort by ProductID. This question will introduce what may be a new concept, the Join clause in SQL. The Join clause is used to join two or more relational database tables together in a logical way. Here’s a data model of the relationship between Products and Suppliers.

### Postgres

```
 SELECT p.product_id, p.product_name, s.company_name
FROM products p
JOIN suppliers s ON p.supplier_id = s.supplier_id
ORDER BY p.product_id;
```

## MongoDB

```
db.products.aggregate([
  {
    $lookup: {
      from: "suppliers",
      localField: "_id",
      foreignField: "_id",
      as: "supplier"
    }
  },
  {
    $unwind: "$supplier"
  },
  {
    $project: {
      ProductID: "$_id",
      ProductName: 1,
      CompanyName: "$supplier.company"
    }
  },
  {
    $sort: {
      ProductID: 1
    }
  }
]);
```

## Neo4J

```
 MATCH (p:Product)-[]-(s:Supplier)
RETURN p.productID, p.productName, s.companyName
ORDER BY p.productID;
```

# 4. In the Customers table, show the total number of customers per Country and City.

## Postgres

```
 SELECT country, city, COUNT(*) AS total_customers
FROM customers
GROUP BY country, city;
```

## MongoDB

```
db.Customers.aggregate([
  {
    $group: {
      _id: {
        Country: "$Country",
        City: "$City"
      },
      TotalCustomers: { $sum: 1 }
    }
  },
  {
    $project: {
      Country: "$_id.Country",
      City: "$_id.City",
      TotalCustomers: 1
    }
  }
]);
```

## Neo4J

```
 MATCH (c:Customer)
RETURN c.country AS country, c.city AS city, count(*) AS TotalCustomers
ORDER BY country, city;
```

# 5. We're doing inventory, and need to show information like the below, for

all orders. Sort by order_id and product_id.

## Postgres

```
 SELECT o.order_id, od.product_id, p.product_name, od.quantity, od.unit_price
FROM orders o
JOIN order_details od ON o.order_id = od.order_id
JOIN products p ON od.product_id = p.product_id
ORDER BY o.order_id, od.product_id;
```

## MongoDB

```
db.orders.aggregate([
  {
    $lookup: {
      from: "orders",
      localField: "_id",
      foreignField: "_id",
      as: "order"
    }
  },
  {
    $unwind: "$order"
  },
  {
    $lookup: {
      from: "products",
      localField: "order._id",
      foreignField: "_id",
      as: "product"
    }
  },
  {
    $unwind: "$product"
  },
  {
    $project: {
      OrderID: 1,
      ProductID: "$order._id",
      ProductName: "$product.product_name",
      Quantity: "$order.quantity_per_unit",
    }
  },
  {
    $sort: {
      OrderID: 1,
      ProductID: 1,
    }
  }
]);
```

### Neo4J

```
 MATCH (o:Order)-[]-(p:Product)
RETURN o.orderID, p.productID, p.productName
ORDER BY o.orderID, p.productID;
```

## 6. Customers with no orders for EmployeeID 4, 1. One employee (Nancy Freehafer, EmployeeID 1) has placed the most orders. However, there are some customers who've never placed an order with her. Show only those customers who have never placed an order with her.

### Postgres

```
 Select
customers.customer_id
,orders.customer_id
From customers
left join orders
on orders.customer_id = customers.customer_id
and orders.employee_id = 4
Where
orders.customer_id is null
```

## MongoDB

```
 db.customers.find({
  _id: { $nin: db.orders.distinct("_id", { _id: 4 }) }
});
```

## Neo4J

```
 MATCH (c:Customer)
WHERE NOT EXISTS {
  MATCH (c)-[:PLACED]->(o:Order)-[:HANDLED_BY]->(:Employee {employeeID: 4})
}
RETURN c.customerID, c.contactName;
```