2(a).

```
-- speedy express is a shipper w/ shipperId 1
-- each order has a shipperId associated with it

-- Assuming that each record in the order table is a distinct order
-
-      I can join the order table to the shipper table through the foreign 'Shipp
erId' key.

-
- Since I only want to get speedy express shipments, I reduce the result set to S
hipperId's of 1
--      (since speedy shippers has a shipping id of 1)

-
- The remaining orders are those for speedy express so use COUNT(*) to count the
number of orders

SELECT COUNT(*) AS TotalOrderShipsSpeedyExpress
FROM Shippers s
JOIN Orders o
    ON s.ShipperId = o.ShipperId
WHERE s.ShipperId = 1

-- Answer: 54 orders
```

2(b).

```
-
- Starting with the employee table, there are EmployeeId and LastNm for 10 employ
ees
-
- Now, looking at the order table, there is a EmployeeId foreign key for each ord
er record

-
- Since the order table contains a foreign key for the employee, I can join both
tables
-
-      together using an inner join on this foreign key. This puts the correspond
ing order and employee data together
```

```
-
- With the employee and order data together, I need to be able to count the numbe
r of records for a certain employeeId.

-
- To do this, I first need to group the data by employee. I can now use COUNT(*)
to count
--      the number of records/orders being grouped by employee

-
- To get the most orders, I should order the records in a descending manner (most
 orders at the top)

SELECT e.LastName, COUNT(*) AS NumOrders
FROM Employees e
JOIN Orders o
    ON e.EmployeeId = o.EmployeeId
GROUP BY e.EmployeeId
ORDER BY COUNT(*) DESC
-- LIMIT 1;

-- Answer: Peacock
```

2(c).

```
-- The customers table contains, PK: CustomerId, Column: country
-
- The products table has, PK: ProductId, Column: product name (answer will be pul
led from here), FK: SupplierId, CategoryId
-- The orders table has, PK: OrderId, FK: CustomerId, EmployeeId, ShipperId
-- NOTE: Orders has customerId FK, cusmoters has country
-
- The orderDetails table has: PK OrderDetailId, FK: OrderId, ProductId, Column: Q
uantity
-
- NOTE: Orders and OrderDetails can be joined to get product information from an
order (in terms of type and quantity)

-
- start by joining the orders with only the german customers to ensure we are onl
y looking at German orders for the rest of the query.
```

```sql
-
-      Join on using the customerId foreign key in the order table (which is the
primary key for customers)

-
- We need most ordered products - the order table does not contain any info about
 qunatity but the OrderDetails does. This means we need to join the OrderDetail t
able
-
-      (again using a the OrderId foreign key in the OrderDetails table) in order
 to have access to specific products and quantities.

-
- The OrderDetails also contains a productId foreign key; I ultimately want produ
ct names so I must join the product table to the OrderDetails table
--      to get access to the product names.

-
- Now that I have orders with associated productIds, productNames and quantities
for German customers, I can group and order the data to solve the problem.
-
- Since I am looking for most popular products, I know I must group the current r
esult-set by productId's so that everything is in terms of products.

-
- Now, I can order the products by the total quantity (which would be the sums of
 the individual qunaities). By ordering the total quantities
-
-      in a descending manner, I can have the most ordered product on top of the
final query result.

-
- Since each productId corresponds with a single ProductName, I can select the pr
oduct name (without any interference from the group by clause)


SELECT p.ProductName, SUM(od.quantity) AS TotalGermanCustomerOrders
FROM Customers c
JOIN Orders o
    ON c.CustomerId = o.CustomerId
        AND c.Country = 'Germany'
JOIN OrderDetails od
    ON od.OrderId = o.OrderId
JOIN Products p
    ON p.ProductId = od.ProductId
```

```sql
GROUP BY p.ProductId
ORDER BY SUM(od.quantity) DESC
-- LIMIT 1;

-- Answer: Boston Crab Meat
```