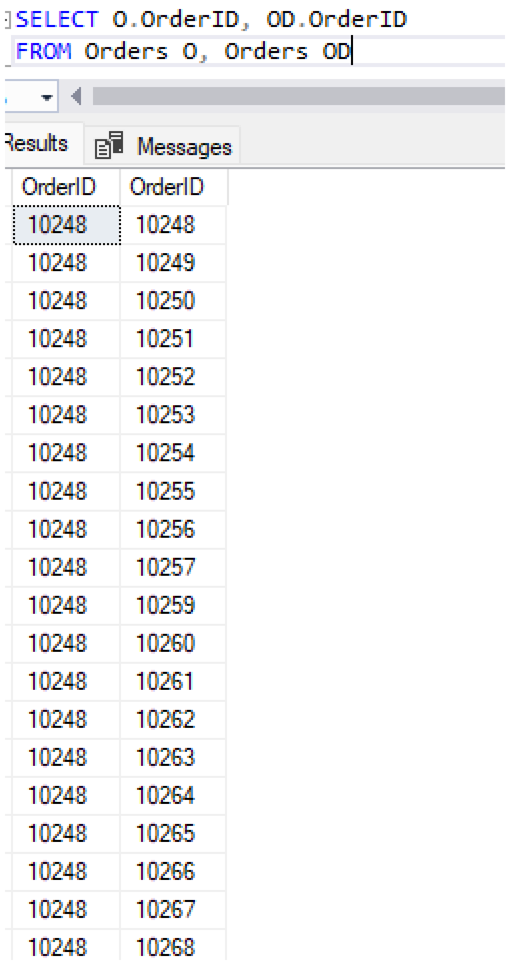
CS262-L **Database Systems**
Spring 2023 **Due:** Week 5

Submitted by: Afraz Butt, 2021-CS-12
Submitted to: Mr. Nauman Babar

Concept of Joins and Applications on Northwind schema

1. Practice different types of joins.

   The different types of joins are self join, cross join, equi join, inner join, left outer join, right outer join, and full outer join.
   Self join is a join of a table with itself.

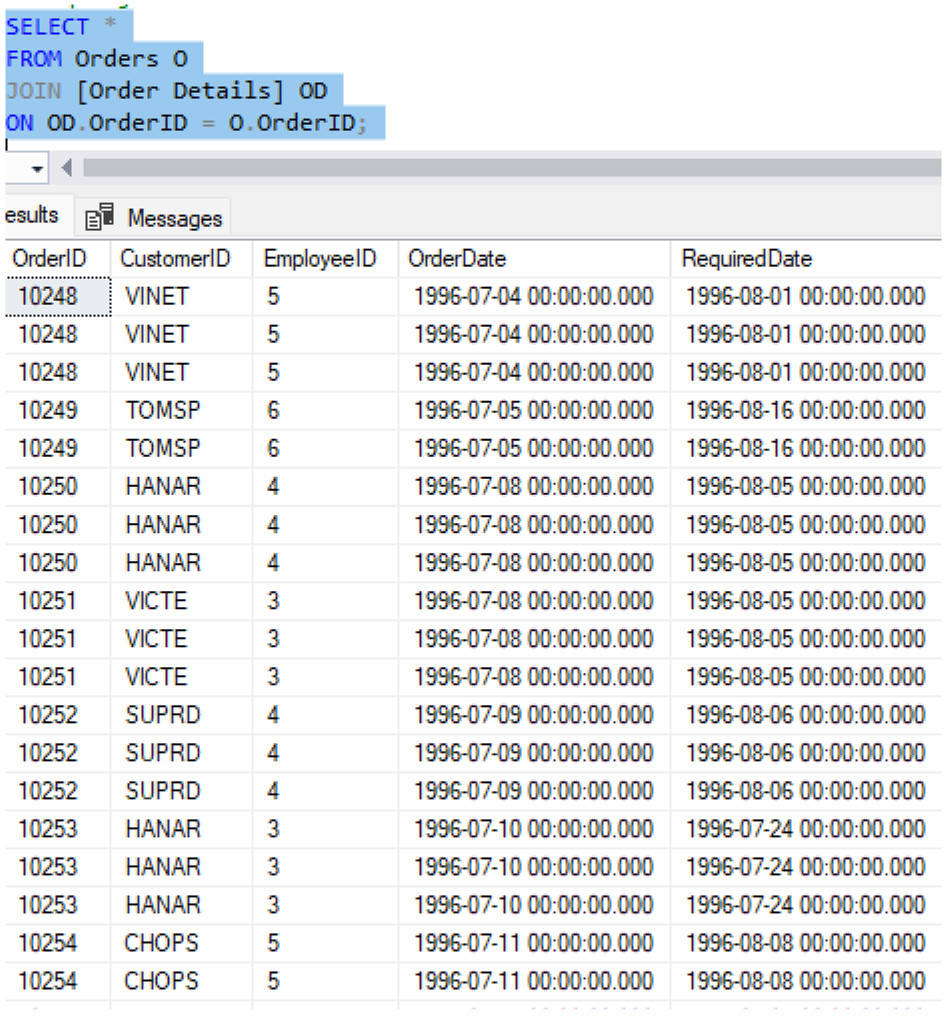


```
SELECT O.OrderID, OD.OrderID
FROM Orders O, Orders OD
```

```
OrderID  OrderID
10248    10248
10248    10249
10248    10250
10248    10251
10248    10252
10248    10253
10248    10254
10248    10255
10248    10256
10248    10257
10248    10259
10248    10260
10248    10261
10248    10262
10248    10263
10248    10264
10248    10265
10248    10266
10248    10267
10248    10268
```

Figure 1: *A query run on Orders table on northwind data. Shows self join and output.*

Equi join is a type of join in where there is a condition specified in the WHERE / HAVING clause of the query. It is equivalent to theta join.



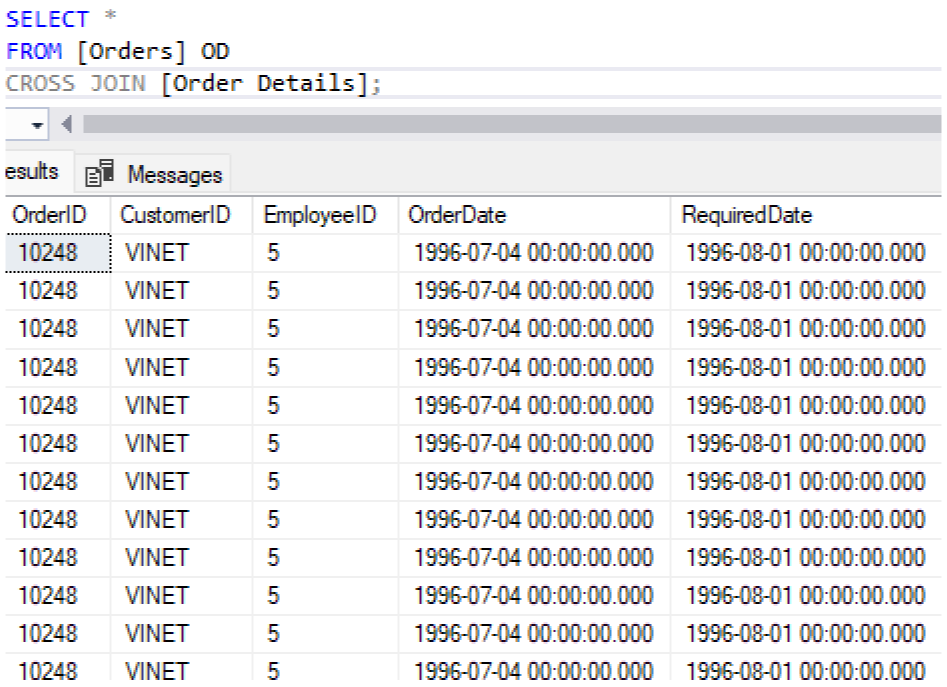
```
SELECT *
FROM Orders O
JOIN [Order Details] OD
ON OD.OrderID = O.OrderID;
```

OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate
10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000
10249	TOMSP	6	1996-07-05 00:00:00.000	1996-08-16 00:00:00.000
10249	TOMSP	6	1996-07-05 00:00:00.000	1996-08-16 00:00:00.000
10250	HANAR	4	1996-07-08 00:00:00.000	1996-08-05 00:00:00.000
10250	HANAR	4	1996-07-08 00:00:00.000	1996-08-05 00:00:00.000
10250	HANAR	4	1996-07-08 00:00:00.000	1996-08-05 00:00:00.000
10251	VICTE	3	1996-07-08 00:00:00.000	1996-08-05 00:00:00.000
10251	VICTE	3	1996-07-08 00:00:00.000	1996-08-05 00:00:00.000
10251	VICTE	3	1996-07-08 00:00:00.000	1996-08-05 00:00:00.000
10252	SUPRD	4	1996-07-09 00:00:00.000	1996-08-06 00:00:00.000
10252	SUPRD	4	1996-07-09 00:00:00.000	1996-08-06 00:00:00.000
10252	SUPRD	4	1996-07-09 00:00:00.000	1996-08-06 00:00:00.000
10253	HANAR	3	1996-07-10 00:00:00.000	1996-07-24 00:00:00.000
10253	HANAR	3	1996-07-10 00:00:00.000	1996-07-24 00:00:00.000
10253	HANAR	3	1996-07-10 00:00:00.000	1996-07-24 00:00:00.000
10254	CHOPS	5	1996-07-11 00:00:00.000	1996-08-08 00:00:00.000
10254	CHOPS	5	1996-07-11 00:00:00.000	1996-08-08 00:00:00.000

Figure 2: *An equi join implemented on a northwind schema.*

In relational databases, a cross join, also known as a Cartesian product, is a type of join operation that returns all possible combinations of rows from two tables.

In a cross join, each row of the first table is combined with every row of the second table, resulting in a result set that has a number of rows equal to the product of the number of rows in each table. For example, if Table A has 5 rows and Table B has 3 rows, a cross join between the two tables would produce a result set with 15 rows (5 x 3).



```
SELECT *
FROM [Orders] OD
CROSS JOIN [Order Details];
```

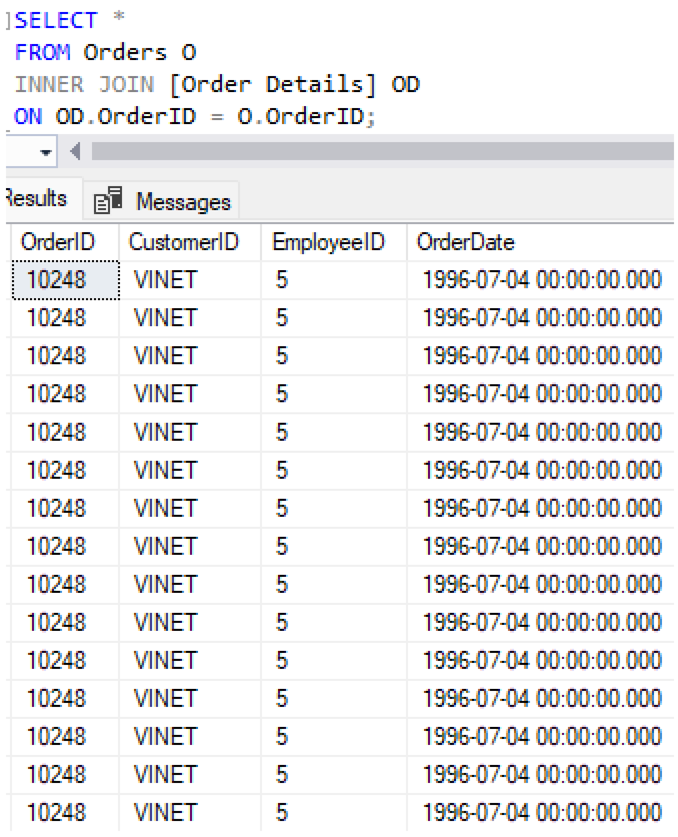
esults | Messages

OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate
10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000

Figure 3: *A cross join implemented on northwind schema.*

An inner join is a type of join operation that returns only the rows that have matching values in both tables being joined.

In other words, an inner join combines rows from two tables based on a common column or set of columns, and only the rows that have matching values in both tables are included in the result set. The resulting table will contain only those rows that satisfy the join condition.



```sql
SELECT *
FROM Orders O
INNER JOIN [Order Details] OD
ON OD.OrderID = O.OrderID;
```

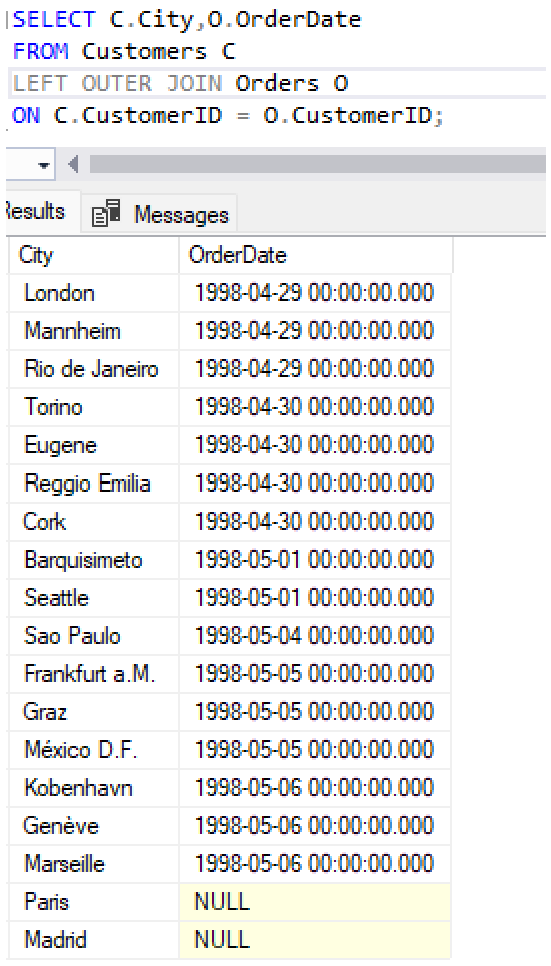
Results | Messages

OrderID	CustomerID	EmployeeID	OrderDate
10248	VINET	5	1996-07-04 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000

Figure 4: *Inner join exhibited. Notice the fact that the order ID for VINET customer is same, meaning a cross join was implemented on Order Details.*

A left outer join (also known as a left join) is a type of join operation that returns all the rows from the left table and only the matching rows from the right table. If there is no match in the right table, the result will still include the row from the left table, but the values for the right table columns will be null.

In other words, a left outer join returns all the rows from the left table and the matching rows from the right table, if any. If there is no match in the right table, the result will still include the row from the left table, but the values for the right table columns will be null.

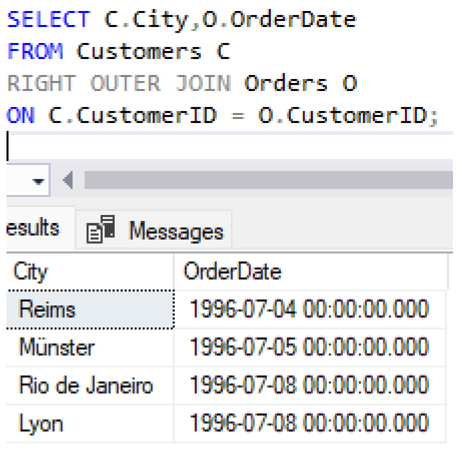


```
SELECT C.City,O.OrderDate
FROM Customers C
LEFT OUTER JOIN Orders O
ON C.CustomerID = O.CustomerID;
```

City	OrderDate
London	1998-04-29 00:00:00.000
Mannheim	1998-04-29 00:00:00.000
Rio de Janeiro	1998-04-29 00:00:00.000
Torino	1998-04-30 00:00:00.000
Eugene	1998-04-30 00:00:00.000
Reggio Emilia	1998-04-30 00:00:00.000
Cork	1998-04-30 00:00:00.000
Barquisimeto	1998-05-01 00:00:00.000
Seattle	1998-05-01 00:00:00.000
Sao Paulo	1998-05-04 00:00:00.000
Frankfurt a.M.	1998-05-05 00:00:00.000
Graz	1998-05-05 00:00:00.000
México D.F.	1998-05-05 00:00:00.000
Kobenhavn	1998-05-06 00:00:00.000
Genève	1998-05-06 00:00:00.000
Marseille	1998-05-06 00:00:00.000
Paris	NULL
Madrid	NULL

Figure 5: *A query and its output exhibited for a left outer join, scrolled down to where a NULL attribute for a non-matching row was present.*

A Right outer join (also known as right join) is the exact opposite of a left outer join.

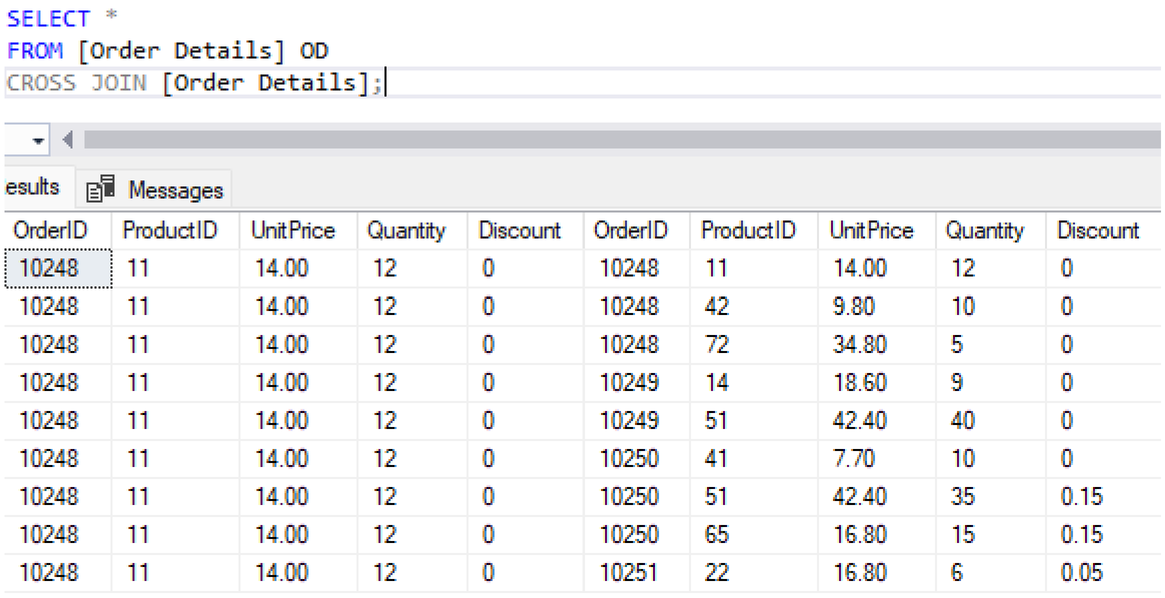


```
SELECT C.City,O.OrderDate
FROM Customers C
RIGHT OUTER JOIN Orders O
ON C.CustomerID = O.CustomerID;
```

City	OrderDate
Reims	1996-07-04 00:00:00.000
Münster	1996-07-05 00:00:00.000
Rio de Janeiro	1996-07-08 00:00:00.000
Lyon	1996-07-08 00:00:00.000

Figure 6: *A query and output for a right outer join exhibited.*

2. Check the difference between a Cross Join and a self-cross join.
   Basically, both are the same. A Cross Join *unconditionally* joins the table having a cartesian product between the rows. While a self-cross join does exactly the same job, but is a *formalized* way of doing so. Below attached are two images, we can see that the output is exactly the same.

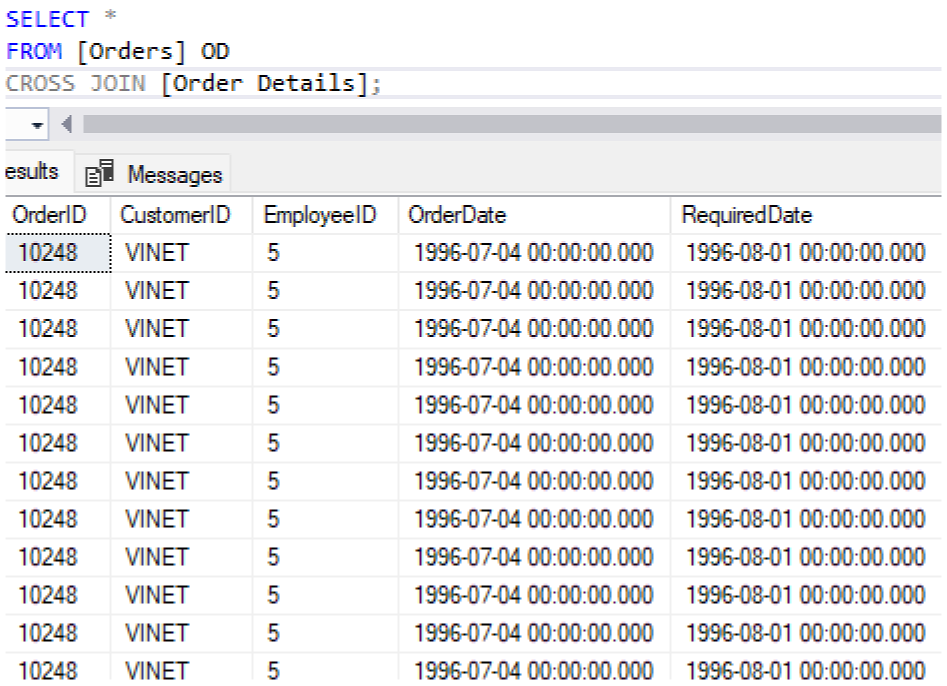


```sql
SELECT *
FROM [Order Details] OD
CROSS JOIN [Order Details];
```

esults | Messages

OrderID	ProductID	UnitPrice	Quantity	Discount	OrderID	ProductID	UnitPrice	Quantity	Discount
10248	11	14.00	12	0	10248	11	14.00	12	0
10248	11	14.00	12	0	10248	42	9.80	10	0
10248	11	14.00	12	0	10248	72	34.80	5	0
10248	11	14.00	12	0	10249	14	18.60	9	0
10248	11	14.00	12	0	10249	51	42.40	40	0
10248	11	14.00	12	0	10250	41	7.70	10	0
10248	11	14.00	12	0	10250	51	42.40	35	0.15
10248	11	14.00	12	0	10250	65	16.80	15	0.15
10248	11	14.00	12	0	10251	22	16.80	6	0.05

Figure 7: *A self cross join on northwind schema.*



```sql
SELECT *
FROM [Orders] OD
CROSS JOIN [Order Details];
```

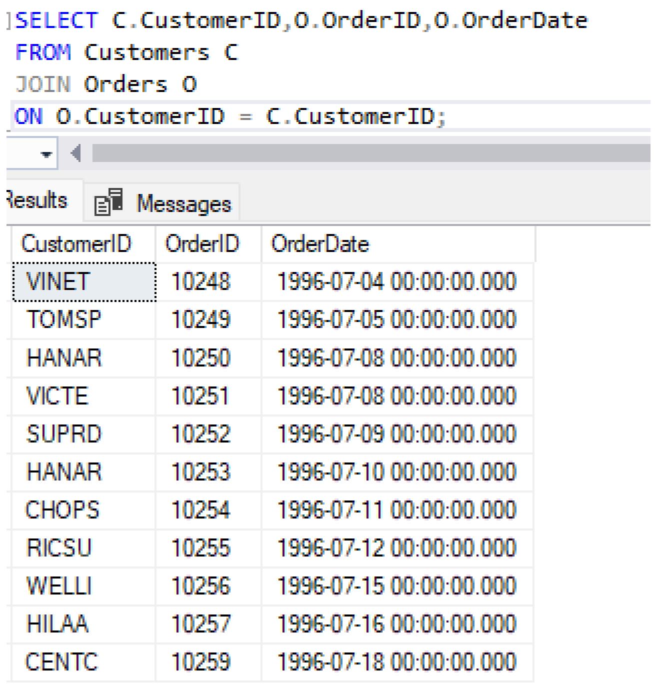
esults | Messages

OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate
10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000
10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000

Figure 8: *Check that the theory for this and cross join in Figure 3 is exactly the same.*

3. Advanced Tasks:

- Report all customers and their orders, who placed no orders.

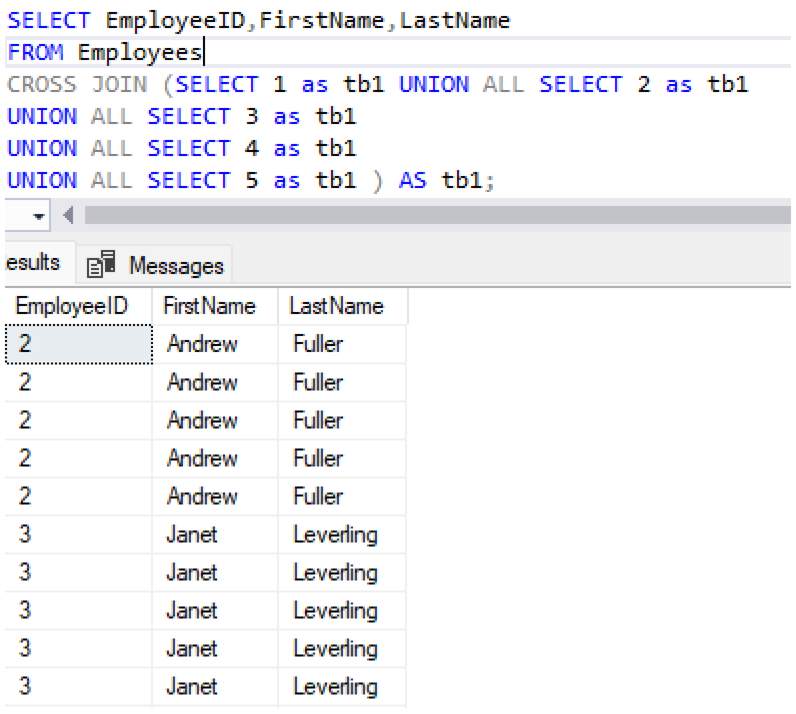


```
SELECT C.CustomerID,O.OrderID,O.OrderDate
FROM Customers C
JOIN Orders O
ON O.CustomerID = C.CustomerID;
```

CustomerID	OrderID	OrderDate
VINET	10248	1996-07-04 00:00:00.000
TOMSP	10249	1996-07-05 00:00:00.000
HANAR	10250	1996-07-08 00:00:00.000
VICTE	10251	1996-07-08 00:00:00.000
SUPRD	10252	1996-07-09 00:00:00.000
HANAR	10253	1996-07-10 00:00:00.000
CHOPS	10254	1996-07-11 00:00:00.000
RICSU	10255	1996-07-12 00:00:00.000
WELLI	10256	1996-07-15 00:00:00.000
HILAA	10257	1996-07-16 00:00:00.000
CENTC	10259	1996-07-18 00:00:00.000

Figure 9: *A query to report all customers and their orders, who placed no orders.*

- Generate 5 copies of each employee.
  See Figure 10 on the next page.



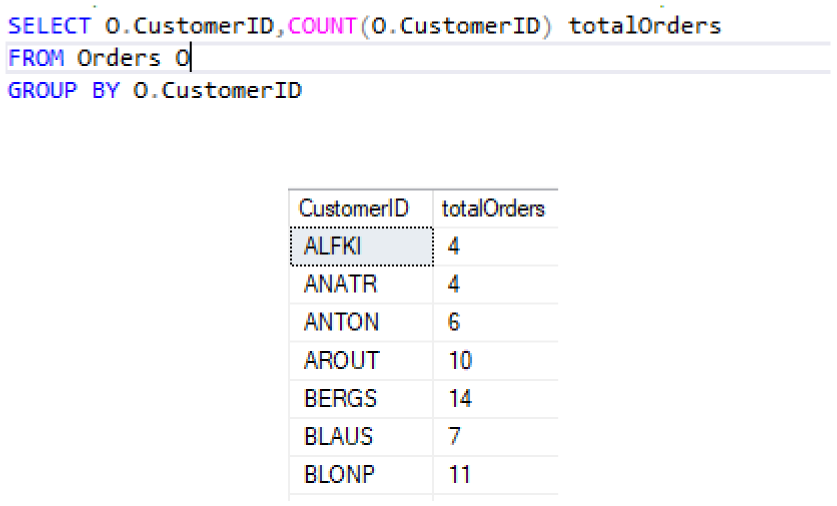
```
SELECT EmployeeID,FirstName,LastName
FROM Employees
CROSS JOIN (SELECT 1 as tb1 UNION ALL SELECT 2 as tb1
UNION ALL SELECT 3 as tb1
UNION ALL SELECT 4 as tb1
UNION ALL SELECT 5 as tb1 ) AS tb1;
```

esults | Messages

EmployeeID	FirstName	LastName
2	Andrew	Fuller
2	Andrew	Fuller
2	Andrew	Fuller
2	Andrew	Fuller
2	Andrew	Fuller
3	Janet	Leverling
3	Janet	Leverling
3	Janet	Leverling
3	Janet	Leverling
3	Janet	Leverling

Figure 10: *A query to generate 5 copies of each employee.*

- Report the total orders of each customer. (customerID, totalorders)

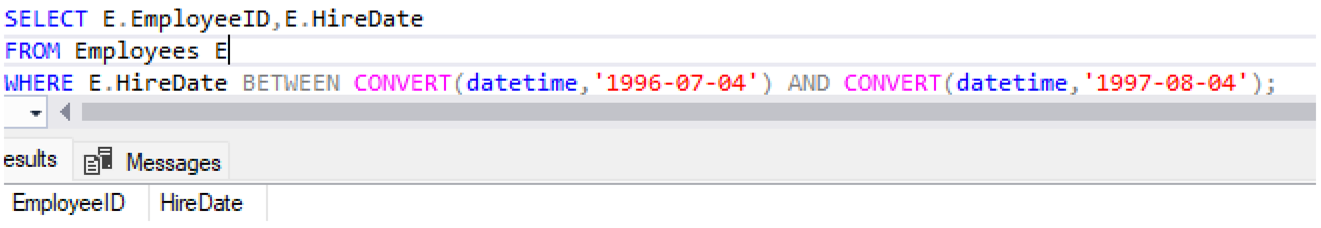


```
SELECT O.CustomerID,COUNT(O.CustomerID) totalOrders
FROM Orders O
GROUP BY O.CustomerID
```

CustomerID	totalOrders
ALFKI	4
ANATR	4
ANTON	6
AROUT	10
BERGS	14
BLAUS	7
BLONP	11

Figure 11: *A query to report the total orders of each customer.*

- Write a query that returns a row for each employee and day in the range 04-07-1996 through 04-08-1997.
  No such employee exists in the table, so our query doesn't return any output.



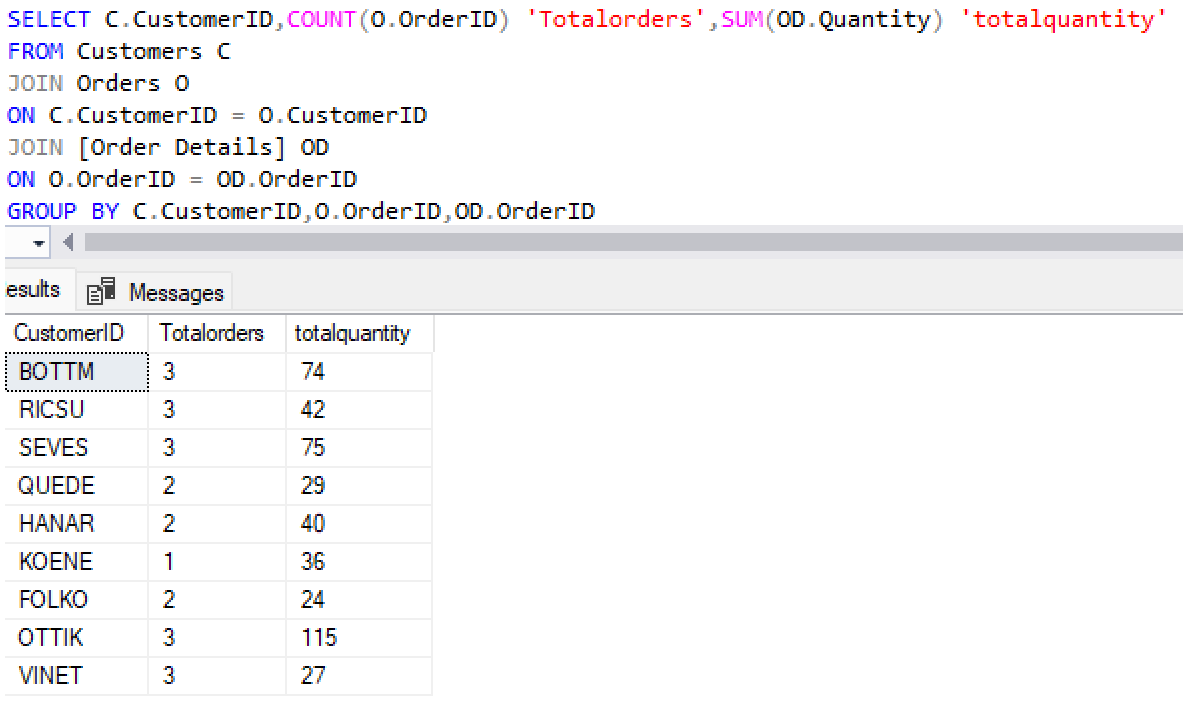
```
SELECT E.EmployeeID,E.HireDate
FROM Employees E
WHERE E.HireDate BETWEEN CONVERT(datetime,'1996-07-04') AND CONVERT(datetime,'1997-08-04');
```

esults | Messages

EmployeeID	HireDate

Figure 12: *A query to returns a row for each employee and day in the range 04-07-1996 through 04-08-1997.*

- Return US customers, and for each customer return the total number of orders and total quantities. (CustomerID, Totalorders, totalquantity).



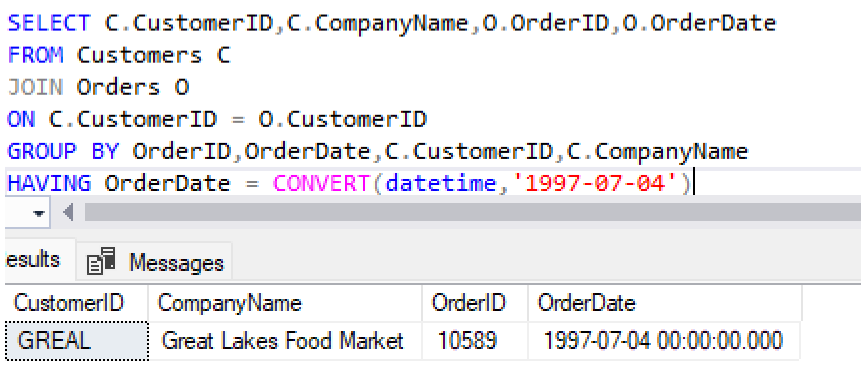
```
SELECT C.CustomerID,COUNT(O.OrderID) 'Totalorders',SUM(OD.Quantity) 'totalquantity'
FROM Customers C
JOIN Orders O
ON C.CustomerID = O.CustomerID
JOIN [Order Details] OD
ON O.OrderID = OD.OrderID
GROUP BY C.CustomerID,O.OrderID,OD.OrderID
```

esults | Messages

CustomerID	Totalorders	totalquantity
BOTTM	3	74
RICSU	3	42
SEVES	3	75
QUEDE	2	29
HANAR	2	40
KOENE	1	36
FOLKO	2	24
OTTIK	3	115
VINET	3	27

Figure 13: *A query to returns US customers, and for each customer return the total number of orders and total quantities.*

- Write a query that returns all customers in the output, but matches them with their respective orders only if they were placed on July 04,1997. (CustomerID, CompanyName, OrderID, Orderdate).

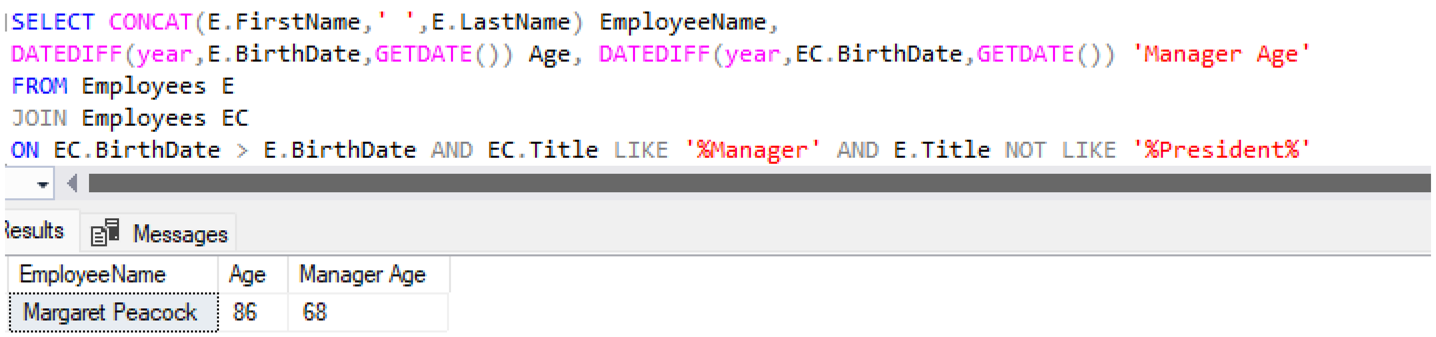


```
SELECT C.CustomerID,C.CompanyName,O.OrderID,O.OrderDate
FROM Customers C
JOIN Orders O
ON C.CustomerID = O.CustomerID
GROUP BY OrderID,OrderDate,C.CustomerID,C.CompanyName
HAVING OrderDate = CONVERT(datetime,'1997-07-04')
```

CustomerID	CompanyName	OrderID	OrderDate
GREAL	Great Lakes Food Market	10589	1997-07-04 00:00:00.000

Figure 14: *A query to return all customers in the output, but matches them with their respective orders only if they were placed on July 04,1997.*

- Are there any employees who are older than their managers? List that names of those employees and their ages. (EmployeeName, Age, Manager Age).

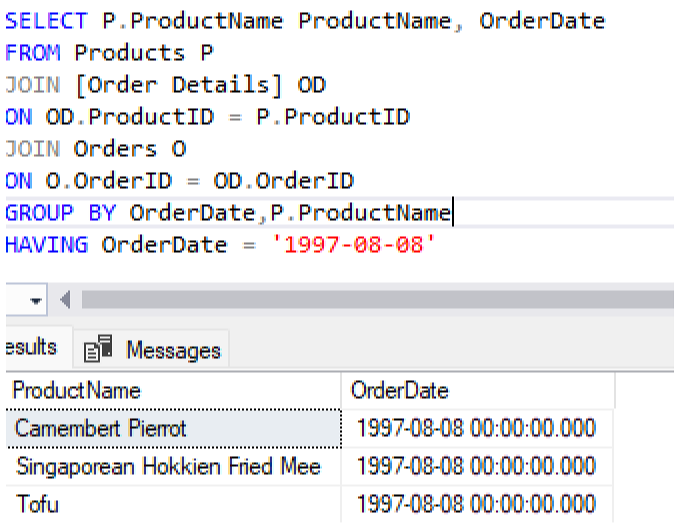


```
SELECT CONCAT(E.FirstName,' ',E.LastName) EmployeeName,
DATEDIFF(year,E.BirthDate,GETDATE()) Age, DATEDIFF(year,EC.BirthDate,GETDATE()) 'Manager Age'
FROM Employees E
JOIN Employees EC
ON EC.BirthDate > E.BirthDate AND EC.Title LIKE '%Manager' AND E.Title NOT LIKE '%President%'
```

EmployeeName	Age	Manager Age
Margaret Peacock	86	68

Figure 15: *A query to list all employees greater than their managers.*

- List the names of products which were ordered on 8th August 1997. (Product-Name, OrderDate).

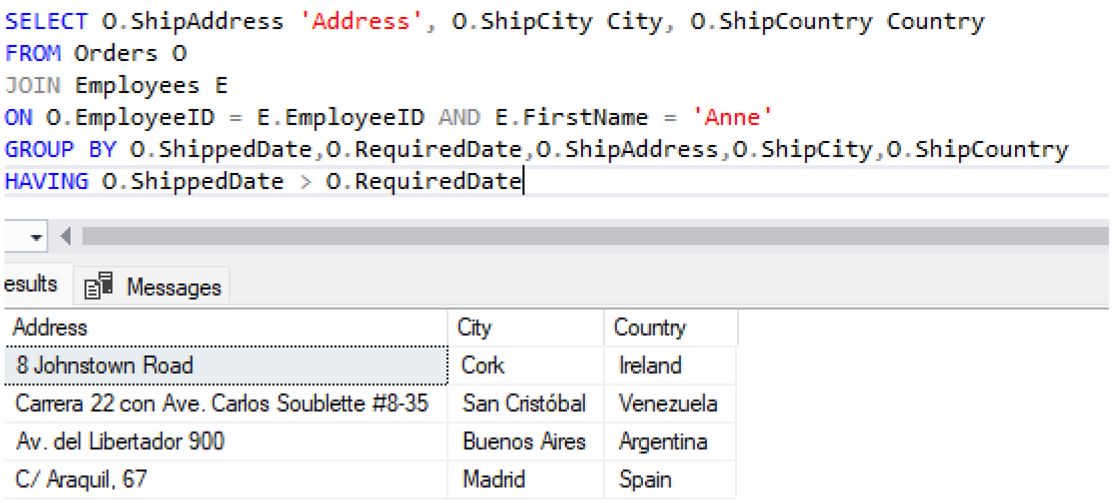


```
SELECT P.ProductName ProductName, OrderDate
FROM Products P
JOIN [Order Details] OD
ON OD.ProductID = P.ProductID
JOIN Orders O
ON O.OrderID = OD.OrderID
GROUP BY OrderDate,P.ProductName
HAVING OrderDate = '1997-08-08'
```

ProductName	OrderDate
Camembert Pierrot	1997-08-08 00:00:00.000
Singaporean Hokkien Fried Mee	1997-08-08 00:00:00.000
Tofu	1997-08-08 00:00:00.000

Figure 16: *A query to list all product names ordered on a given date.*

- List the addresses, cities, countries of all orders which were serviced by Anne and were shipped late. (Address, City, Country).

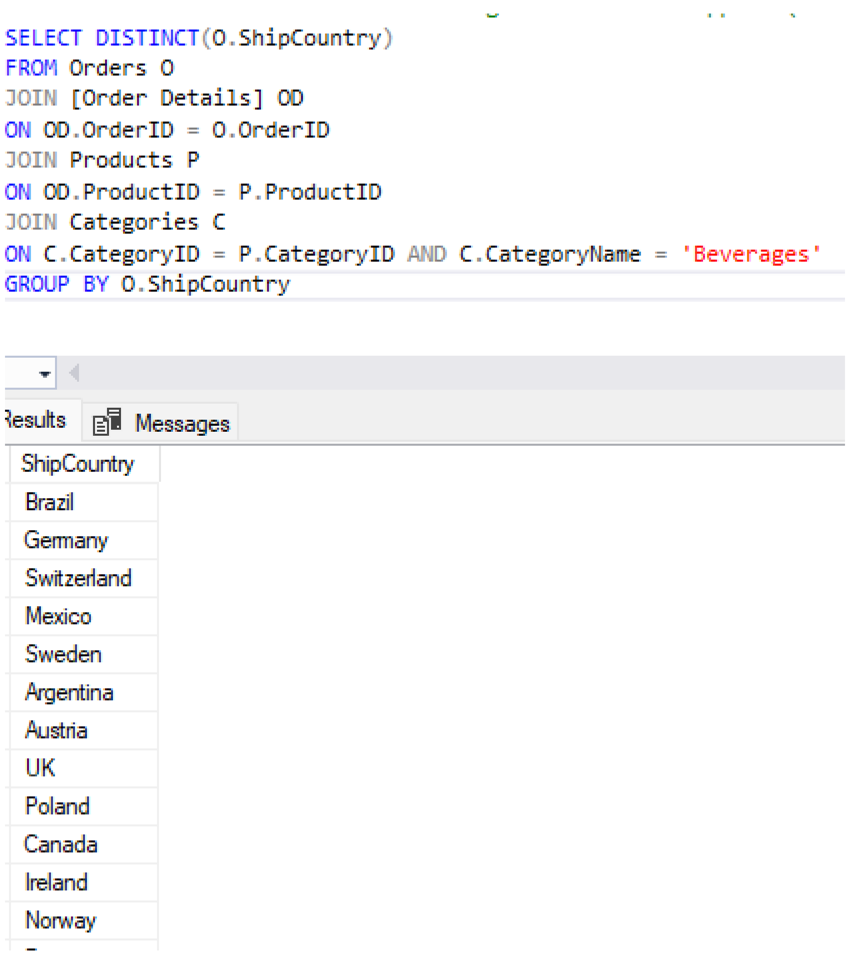


```
SELECT O.ShipAddress 'Address', O.ShipCity City, O.ShipCountry Country
FROM Orders O
JOIN Employees E
ON O.EmployeeID = E.EmployeeID AND E.FirstName = 'Anne'
GROUP BY O.ShippedDate,O.RequiredDate,O.ShipAddress,O.ShipCity,O.ShipCountry
HAVING O.ShippedDate > O.RequiredDate
```

Address	City	Country
8 Johnstown Road	Cork	Ireland
Carrera 22 con Ave. Carlos Soublette #8-35	San Cristóbal	Venezuela
Av. del Libertador 900	Buenos Aires	Argentina
C/ Araquil, 67	Madrid	Spain

Figure 17: *All orders serviced by Anne who were shipped late.*

- List all countries to which beverages have been shipped. (Country).



```
SELECT DISTINCT(O.ShipCountry)
FROM Orders O
JOIN [Order Details] OD
ON OD.OrderID = O.OrderID
JOIN Products P
ON OD.ProductID = P.ProductID
JOIN Categories C
ON C.CategoryID = P.CategoryID AND C.CategoryName = 'Beverages'
GROUP BY O.ShipCountry
```

```
ShipCountry
Brazil
Germany
Switzerland
Mexico
Sweden
Argentina
Austria
UK
Poland
Canada
Ireland
Norway
```

Figure 18: *All orders to which beverages have been shipped.*