UTS:ENGINEERING AND INFORMATION TECHNOLOGY

UTS UNIVERSITY OF TECHNOLOGY, SYDNEY

lecture 8: SQL II
Multiple Table Queries

Main Reference:

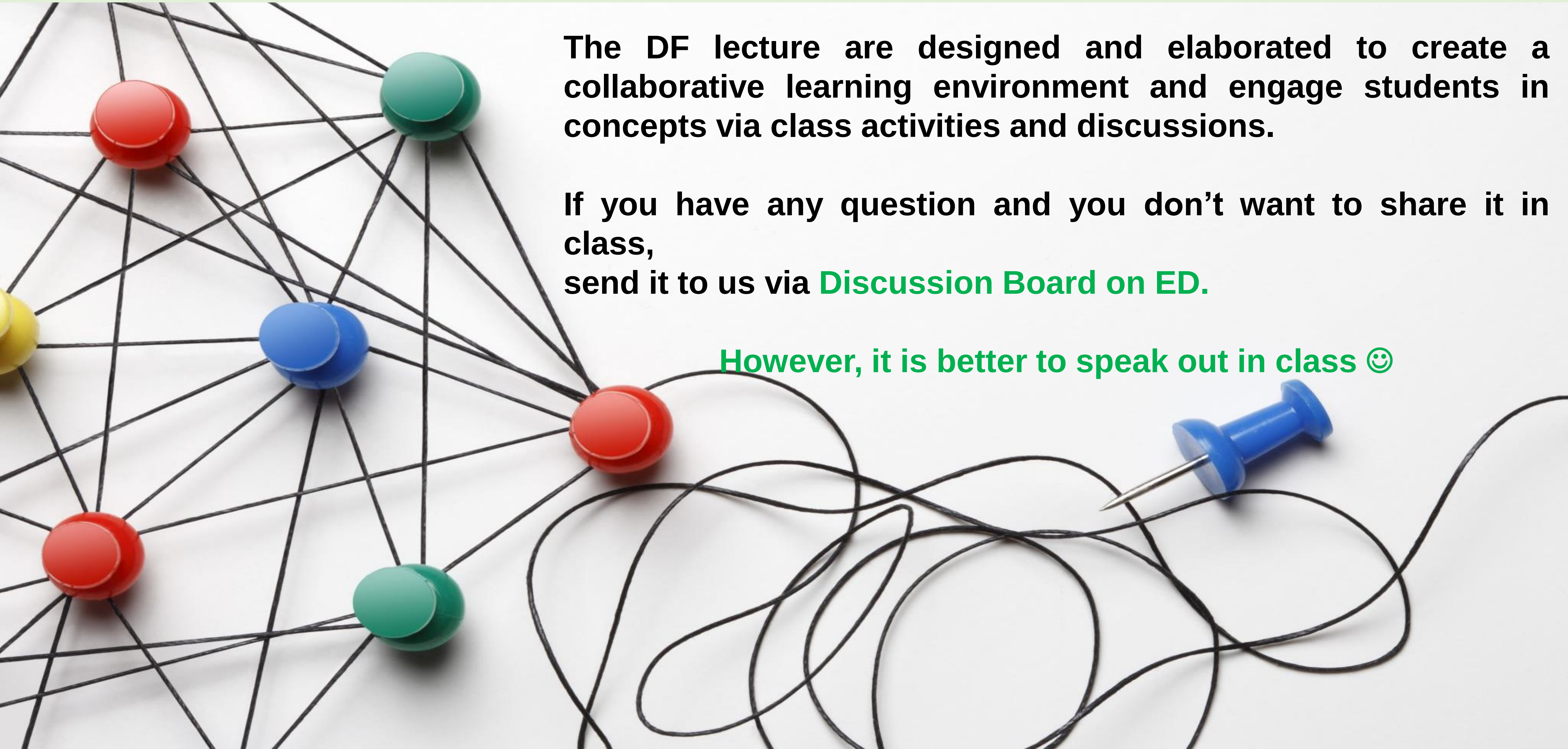
Modern Database Management, 11th Edition
Chapter 7: Advanced SQL

Subject Coordinator and Instructor:

Dr. Danna (Fahimeh) Ramezani

Innovation in practice
eng.uts.edu.au • it.uts.edu.au

Participations and Discussions

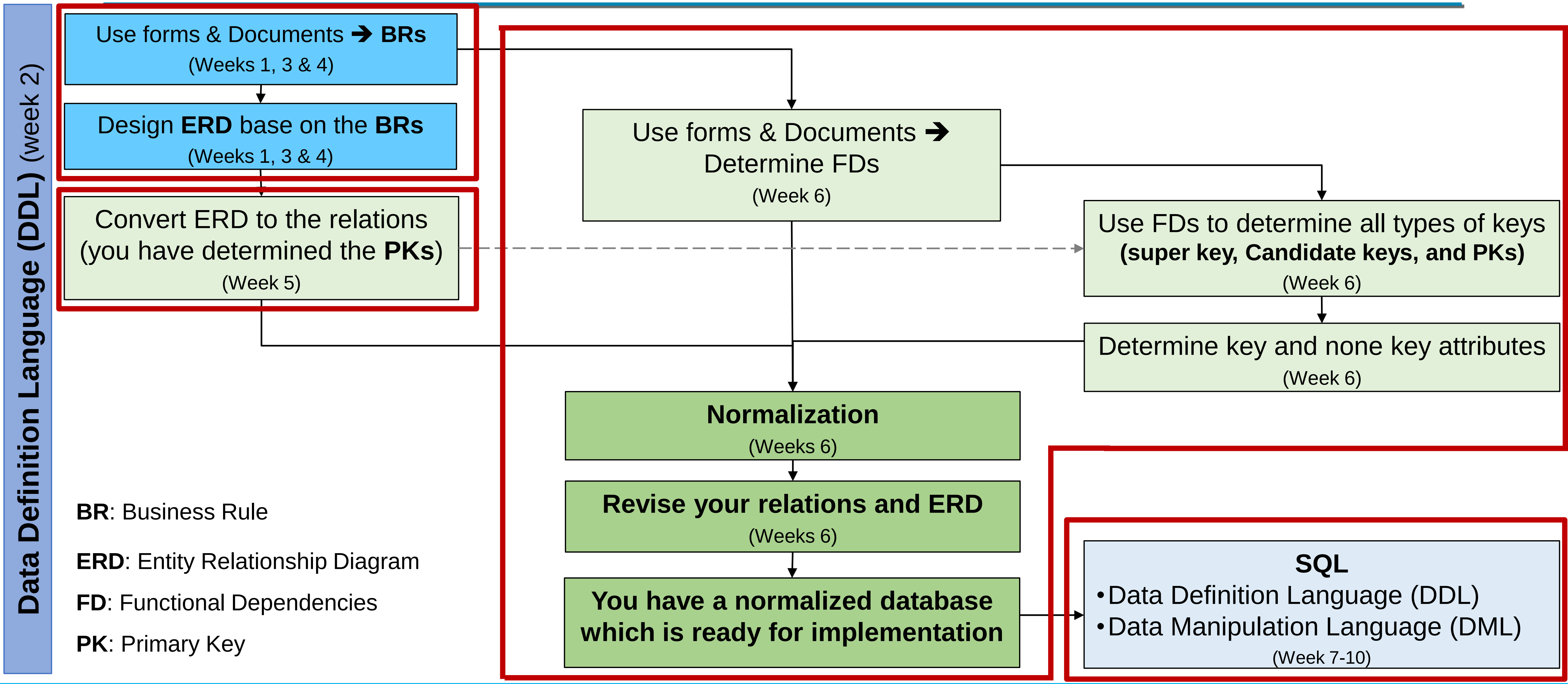


The DF lecture are designed and elaborated to create a collaborative learning environment and engage students in concepts via class activities and discussions.

If you have any question and you don’t want to share it in class,
send it to us via Discussion Board on ED.

However, it is better to speak out in class ☺

Subject Flowchart


Data Definition Language (DDL) (week 2)
Use forms & Documents → BRs (Weeks 1, 3 & 4)
Design ERD base on the BRs (Weeks 1, 3 & 4)
Convert ERD to the relations (you have determined the PKs) (Week 5)
Use forms & Documents → Determine FDs (Week 6)
Use FDs to determine all types of keys (super key, Candidate keys, and PKs) (Week 6)
Determine key and none key attributes (Week 6)
Normalization (Weeks 6)
Revise your relations and ERD (Weeks 6)
You have a normalized database which is ready for implementation
SQL
•Data Definition Language (DDL)
•Data Manipulation Language (DML)
(Week 7-10)

BR: Business Rule

ERD: Entity Relationship Diagram

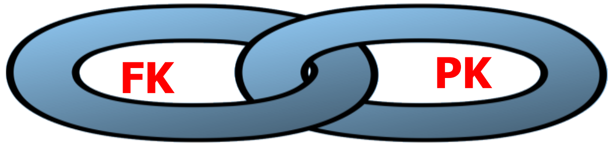
FD: Functional Dependencies

PK: Primary Key

Subject Overview

- **Design Entity Relationship Diagram (ERD)**
  - **Week 1: Data Modelling I (Conceptual Level):** Entity, Attributes, PK, FK, …
  - **Week 2: Data Definition Language (DDL):** Create tables, constraints, insert, …
  - **Week 3: Data Modelling II (Conceptual Level):** Associative, Weak, …
  - **Week 4: Data Modelling III (Conceptual Level):** Subtype/Supertype
  - **Week 5: Convert ERD to Relations (Logical Level)**
  - **Week 6: Functional Dependencies, and Normalization**
- **Data manipulation**
  - **Week 7: Simple Query**
  - **Week 8: Multiple Table Queries**
  - **Week 9: Subquery**
  - **Week 10: Correlated Subquery**

Question: I need the information about **my life** and **my success** after **COVID-19 is gone.**



MySuccess_T

SuccessID	SuccessName	SuccessDate	HappinessID
1967	Got HD Grade in PF	8/10/2019	1755
2055	Got HD Grade in DF	null	1755
3798	Start my job in NASA	null	1899
…	…	…	…

MyLife_T

HappinessID	HappinessName	HppinessStartDate	HppinessEndDate	COVID_19
1755	Pass DF	09/03/2020	null	Gone
1899	Graduated	09/03/2019	null	Came
…	…	…	…	…

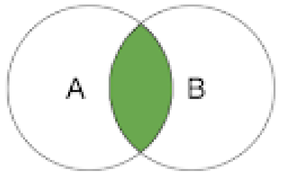
```
Select *
from MySuccess_T Inner Join MyLife_T
on  MySuccess_T. HappinessID = MyLife_T.HappinessID
where COVID_19 = 'Gone';
```

SuccessID	SuccessName	SuccessDate	HappinessID	HappinessID	HappinessName	HppinessStartDate	HppinessEndDate	COVID_19
1967	Got HD Grade in PF	8/10/2019	1755	1755	Pass DF	09/03/2020	null	Gone
2055	Got HD Grade in DF	null	1755	1755	Pass DF	09/03/2020	null	Gone
3798	Start my job in NASA	null	1899	1899	Graduated	09/03/2019	null	Came

Lecture Objectives: JOINS

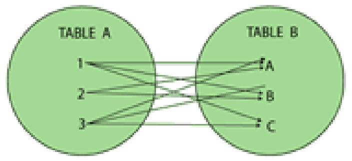
The different join types are visualized with results returned in shaded area

1. Inner join



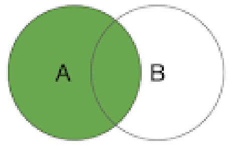
INNER JOIN

2. Cross Join

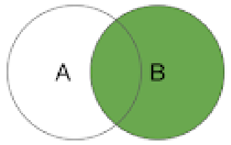


Cross JOIN

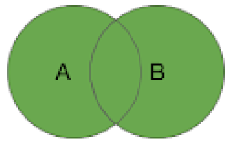
3. Outer join (left / right / full)



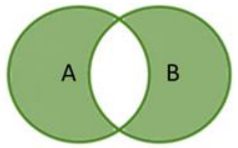
LEFT OUTER JOIN



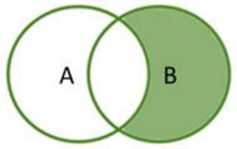
RIGHT OUTER JOIN



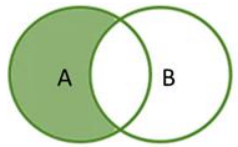
FULL OUTER JOIN



Outer Minus Join

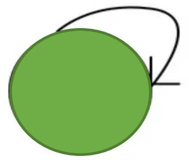


Right Minus Join



Left Minus Join

4. Self join



Self Join

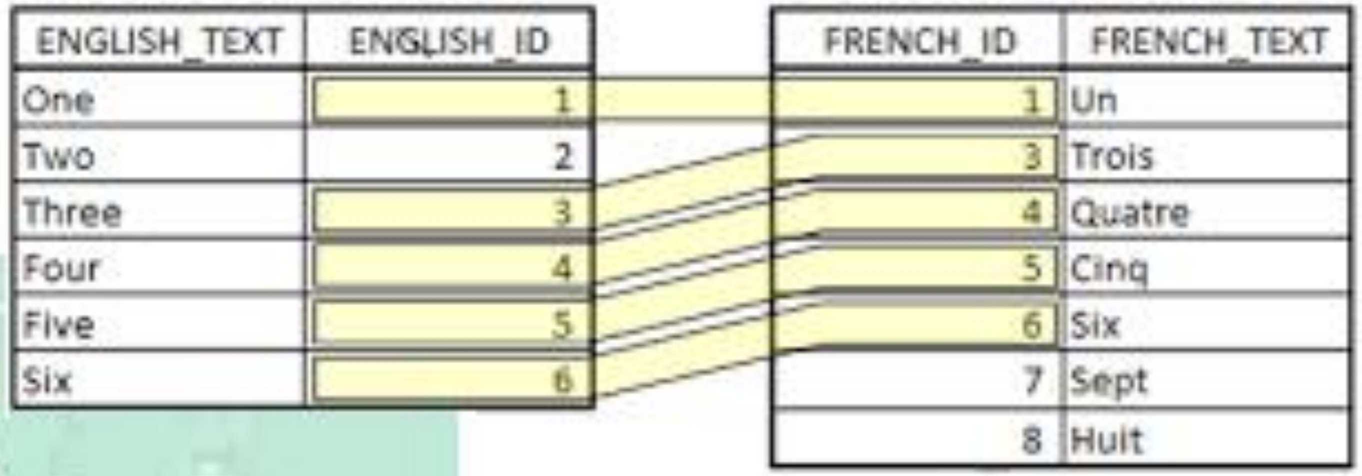
5. Extra information

1. Natural join (don't use this)
2. Unions

Join Multiple Tables

- **Join**: A relational operation that causes **two or more tables** with a **common column** to be combined into a single table or view.
- **Joins involve multiple tables in FROM clause**
- Joins display output from **two or more tables** by finding **matching row values** in **columns that HAVE THE SAME DATA TYPE**.

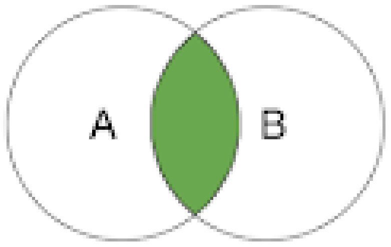
Note: The common columns in joined tables are usually the primary key of the dominant table and the foreign key of the dependent table in 1:M relationships.



ENGLISH_TEXT	ENGLISH_ID
One	1
Two	2
Three	3
Four	4
Five	5
Six	6

FRENCH_ID	FRENCH_TEXT
1	Un
3	Trois
4	Quatre
5	Cinq
6	Six
7	Sept
8	Huit

1. Inner Join



INNER JOIN

1. Type of Join: Inner join

- **Inner Join:** a join in which the joining condition is based on **equality between** values in the common columns; common columns appear redundantly in the result table.

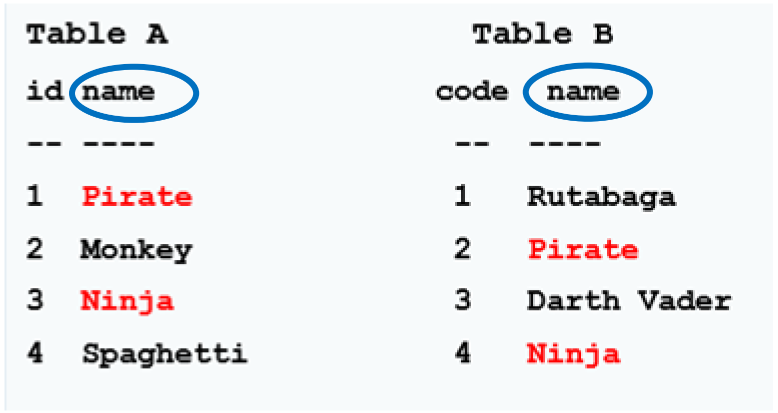


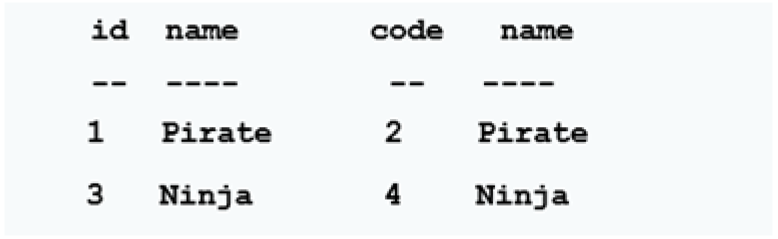
Table A

id	name
--	----
1	Pirate
2	Monkey
3	Ninja
4	Spaghetti

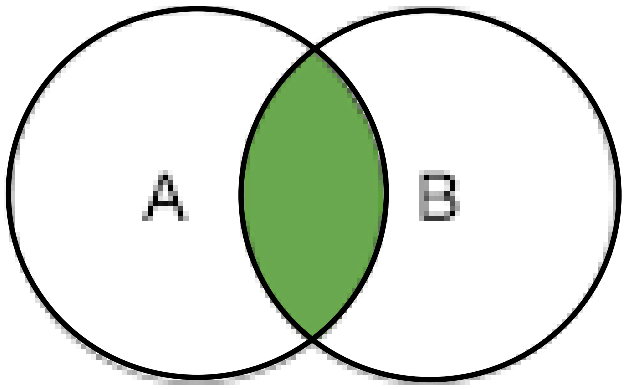
Table B

code	name
--	----
1	Rutabaga
2	Pirate
3	Darth Vader
4	Ninja

```
SELECT *
   FROM TableA INNER JOIN TableB
   ON TableA.name = TableB.name
```

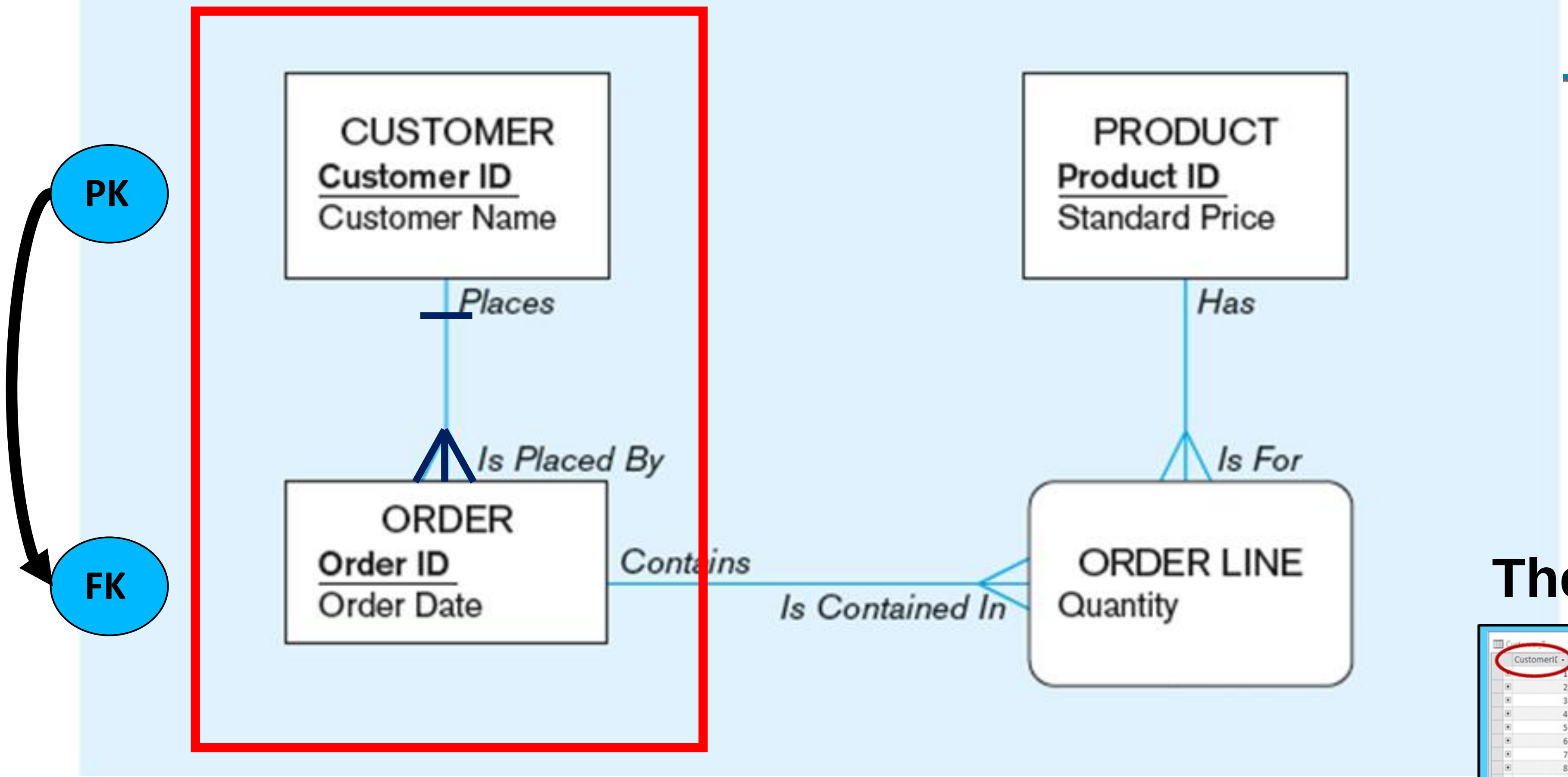


id	name	code	name
--	----	--	----
1	Pirate	2	Pirate
3	Ninja	4	Ninja

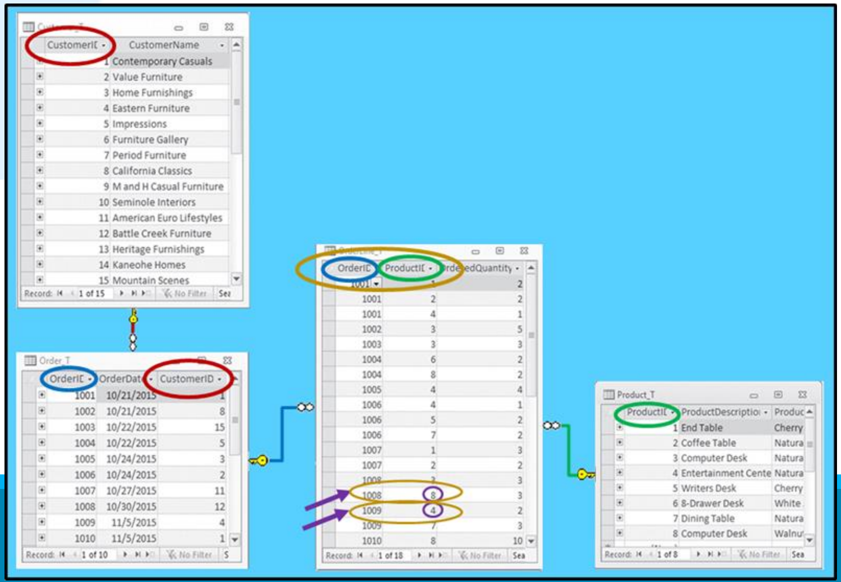


INNER JOIN

The ERD



The Tables



Customer_T

CustomerID	CustomerName
1	Contemporary Casuals
2	Value Furniture
3	Home Furnishings
4	Eastern Furniture
5	Impressions
6	Furniture Gallery
7	Period Furniture
8	California Classics
9	M and H Casual Furniture
10	Seminole Interiors
11	American Euro Lifestyles
12	Battle Creek Furniture
13	Heritage Furnishings
14	Kaneohe Homes
15	Mountain Scenes

Record: 1 of 15

Order_T

OrderID	OrderDate	CustomerID
1001	10/21/2015	1
1002	10/21/2015	8
1003	10/22/2015	15
1004	10/22/2015	5
1005	10/24/2015	3
1006	10/24/2015	2
1007	10/27/2015	11
1008	10/30/2015	12
1009	11/5/2015	4
1010	11/5/2015	1

Record: 1 of 10

OrderLine_T

OrderID	ProductID	OrderedQuantity
1001	1	2
1001	2	2
1001	4	1
1002	3	5
1003	3	3
1004	6	2
1004	8	2
1005	4	4
1006	4	1
1006	5	2
1006	7	2
1007	1	3
1007	2	2
1008	3	3
1008	8	3
1009	4	2
1009	7	3
1010	8	10

Record: 1 of 18

Product_T

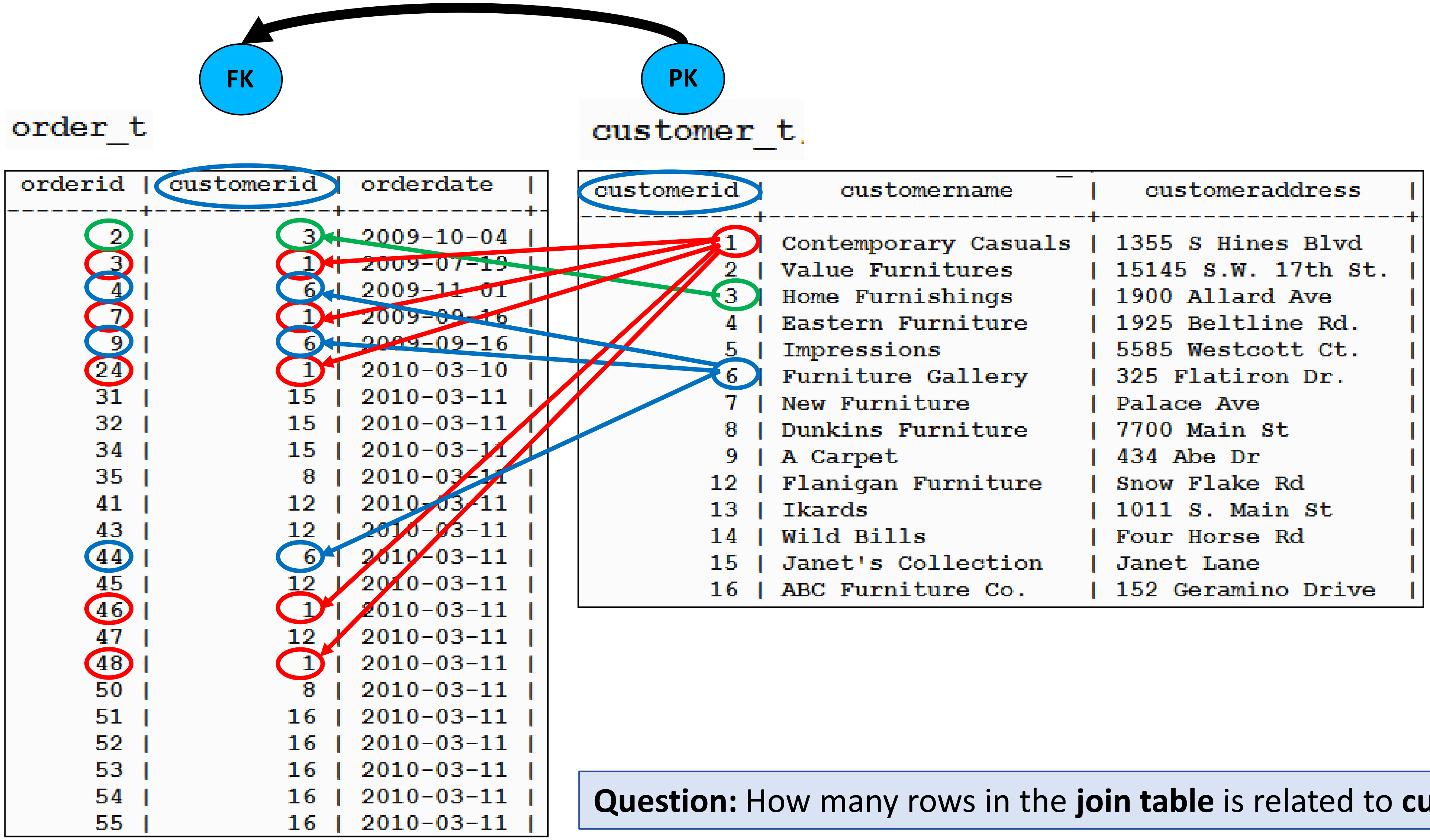
ProductID	ProductDescription	Produc
1	End Table	Cherry
2	Coffee Table	Natura
3	Computer Desk	Natura
4	Entertainment Cente	Natura
5	Writers Desk	Cherry
6	8-Drawer Desk	White
7	Dining Table	Natura
8	Computer Desk	Walnu

Record: 1 of 8

Remember the facts related to a PK/FK pair:

- **PK and FK have the same data type, and**
- **every FK value refer back to corresponding PK value.**

The tables Corresponding to Customer and Order Entities

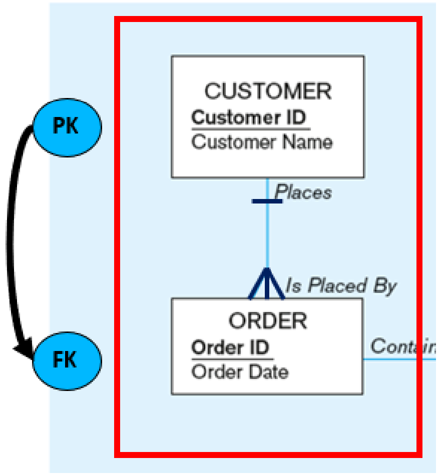


order_t

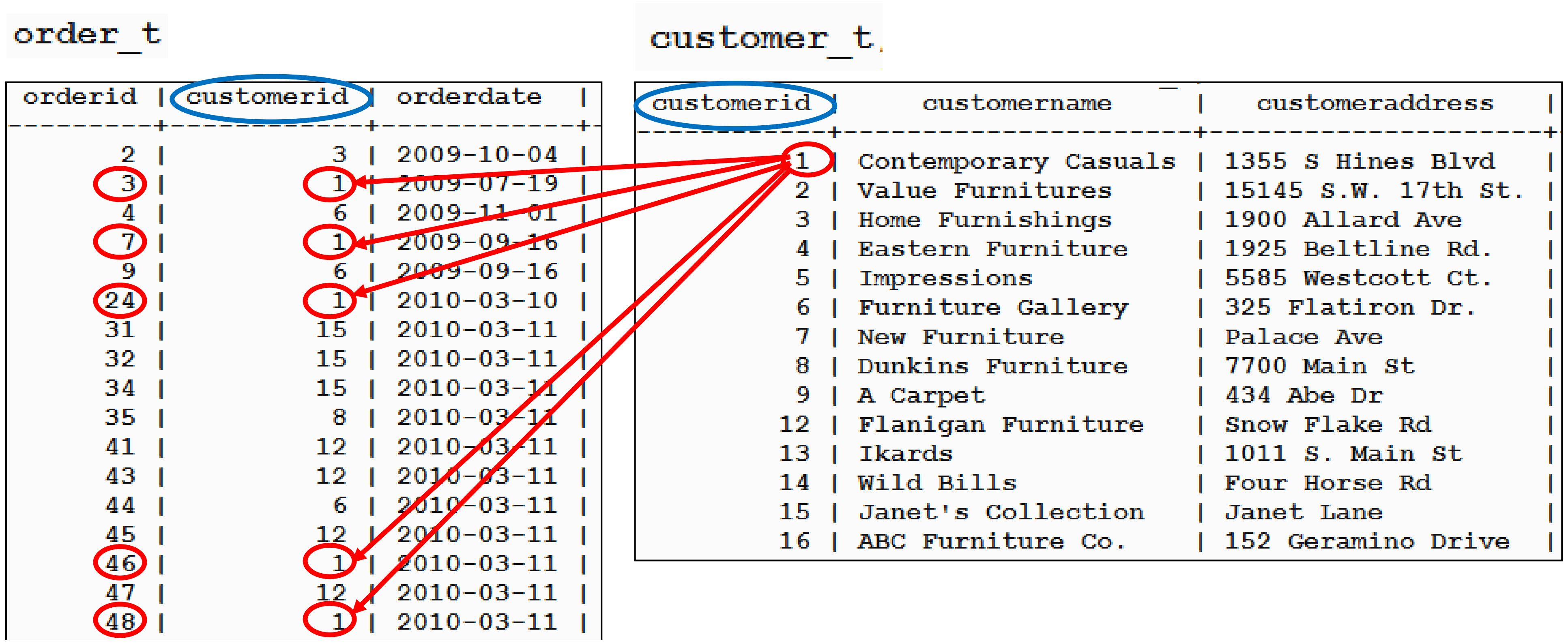
orderid	customerid	orderdate
2	3	2009-10-04
3	1	2009-07-19
4	6	2009-11-01
7	1	2009-09-16
9	6	2009-09-16
24	1	2010-03-10
31	15	2010-03-11
32	15	2010-03-11
34	15	2010-03-11
35	8	2010-03-11
41	12	2010-03-11
43	12	2010-03-11
44	6	2010-03-11
45	12	2010-03-11
46	1	2010-03-11
47	12	2010-03-11
48	1	2010-03-11
50	8	2010-03-11
51	16	2010-03-11
52	16	2010-03-11
53	16	2010-03-11
54	16	2010-03-11
55	16	2010-03-11

customer_t

customerid	customername	customeraddress
1	Contemporary Casuals	1355 S Hines Blvd
2	Value Furnitures	15145 S.W. 17th St.
3	Home Furnishings	1900 Allard Ave
4	Eastern Furniture	1925 Beltline Rd.
5	Impressions	5585 Westcott Ct.
6	Furniture Gallery	325 Flatiron Dr.
7	New Furniture	Palace Ave
8	Dunkins Furniture	7700 Main St
9	A Carpet	434 Abe Dr
12	Flanigan Furniture	Snow Flake Rd
13	Ikards	1011 S. Main St
14	Wild Bills	Four Horse Rd
15	Janet's Collection	Janet Lane
16	ABC Furniture Co.	152 Geramino Drive



Question: How many rows in the **join table** is related to **customer number 1**?

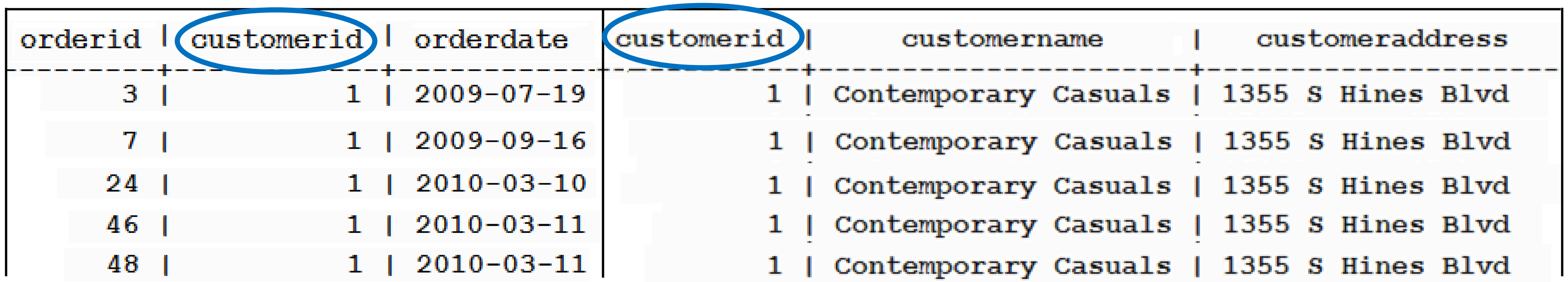


```
order_t

 orderid | customerid | orderdate  |
---------+------------+------------+
       2 |          3 | 2009-10-04 |
       3 |          1 | 2009-07-19 |
       4 |          6 | 2009-11-01 |
       7 |          1 | 2009-09-16 |
       9 |          6 | 2009-09-16 |
      24 |          1 | 2010-03-10 |
      31 |         15 | 2010-03-11 |
      32 |         15 | 2010-03-11 |
      34 |         15 | 2010-03-11 |
      35 |          8 | 2010-03-11 |
      41 |         12 | 2010-03-11 |
      43 |         12 | 2010-03-11 |
      44 |          6 | 2010-03-11 |
      45 |         12 | 2010-03-11 |
      46 |          1 | 2010-03-11 |
      47 |         12 | 2010-03-11 |
      48 |          1 | 2010-03-11 |
```

```
customer_t

 customerid |     customername      |   customeraddress    |
------------+-----------------------+----------------------+
          1 | Contemporary Casuals  | 1355 S Hines Blvd    |
          2 | Value Furnitures      | 15145 S.W. 17th St.  |
          3 | Home Furnishings      | 1900 Allard Ave      |
          4 | Eastern Furniture     | 1925 Beltline Rd.    |
          5 | Impressions           | 5585 Westcott Ct.    |
          6 | Furniture Gallery     | 325 Flatiron Dr.     |
          7 | New Furniture         | Palace Ave           |
          8 | Dunkins Furniture     | 7700 Main St         |
          9 | A Carpet              | 434 Abe Dr           |
         12 | Flanigan Furniture    | Snow Flake Rd        |
         13 | Ikards                | 1011 S. Main St      |
         14 | Wild Bills            | Four Horse Rd        |
         15 | Janet's Collection    | Janet Lane           |
         16 | ABC Furniture Co.     | 152 Geramino Drive   |
```

Question: How many rows in the **join table** is related to **customer number 1**?

```
select *  from order_t inner join customer_t  on order_t.customerid = customer_t.customerid
order by orderid;
```



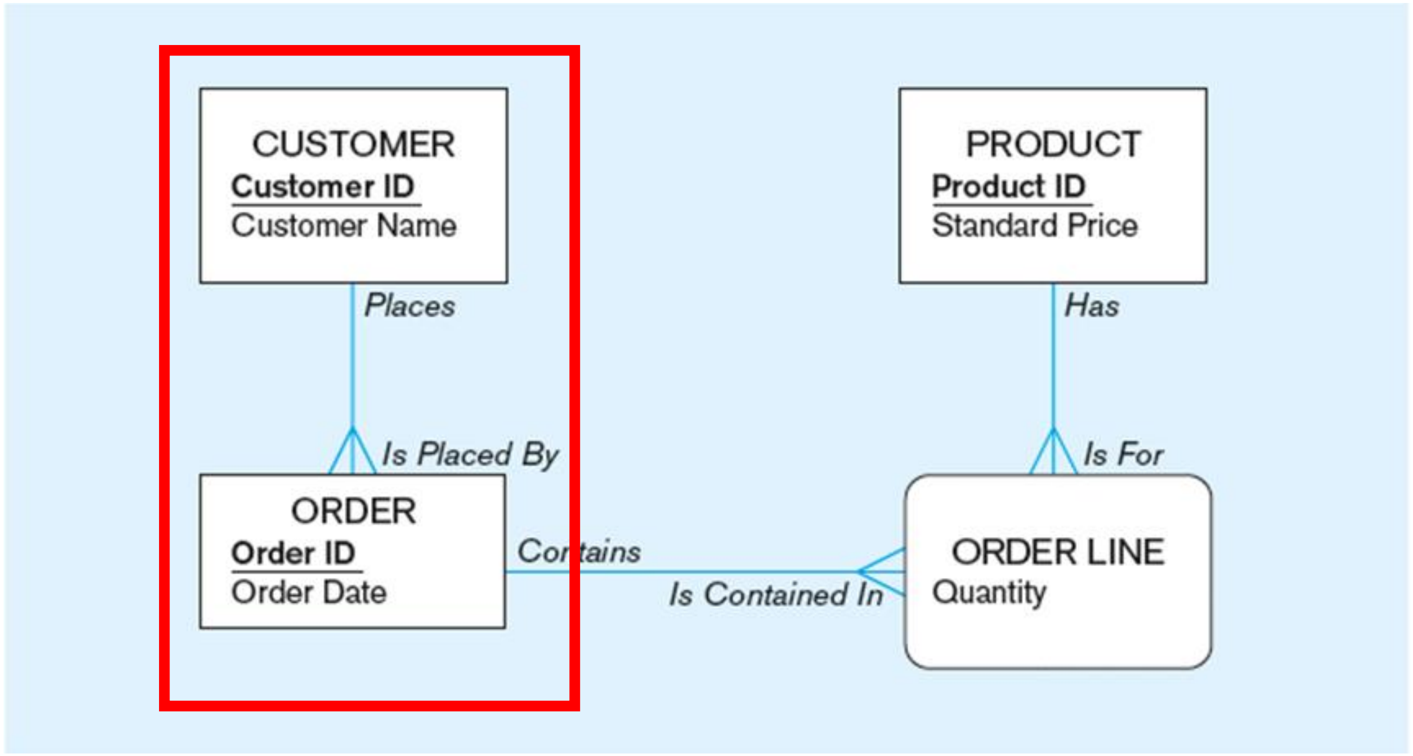
```
 orderid | customerid | orderdate  | customerid |     customername      |  customeraddress   
---------+------------+------------+------------+-----------------------+--------------------
       3 |          1 | 2009-07-19 |          1 | Contemporary Casuals | 1355 S Hines Blvd
       7 |          1 | 2009-09-16 |          1 | Contemporary Casuals | 1355 S Hines Blvd
      24 |          1 | 2010-03-10 |          1 | Contemporary Casuals | 1355 S Hines Blvd
      46 |          1 | 2010-03-11 |          1 | Contemporary Casuals | 1355 S Hines Blvd
      48 |          1 | 2010-03-11 |          1 | Contemporary Casuals | 1355 S Hines Blvd
```

Class Activity 8.1: Inner join

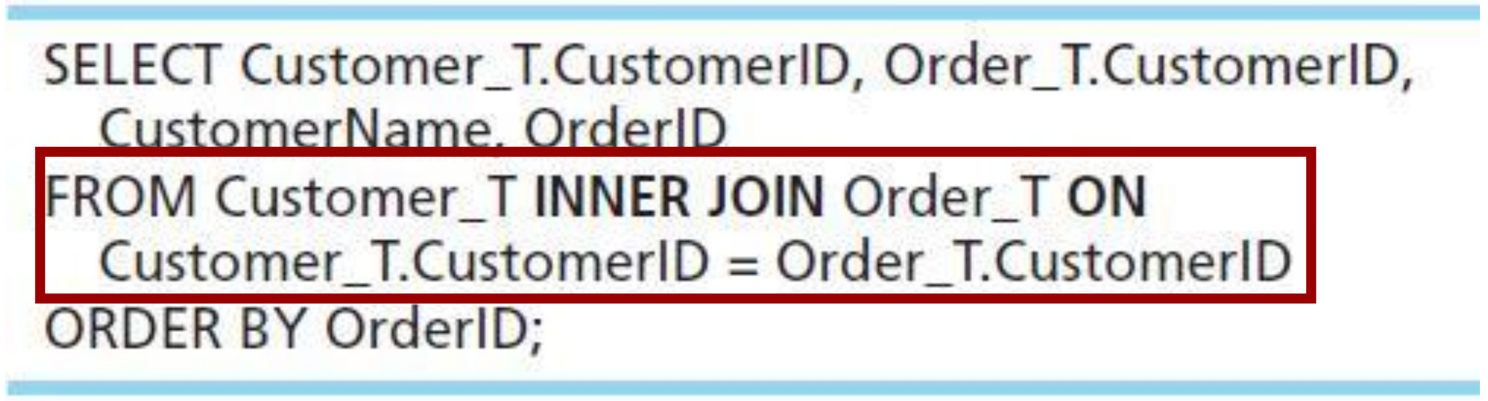
- Determine customers' ID and Name, and their Order ID using inner join.

Inner join query format:

```
SELECT *
FROM TableA INNER JOIN TableB
        ON TableA.name = TableB.name
```



Example 1: Inner join

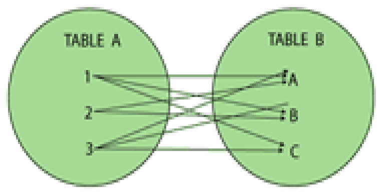


```
SELECT Customer_T.CustomerID, Order_T.CustomerID,
   CustomerName, OrderID
FROM Customer_T INNER JOIN Order_T ON
   Customer_T.CustomerID = Order_T.CustomerID
ORDER BY OrderID;
```

INNER JOIN clause is an alternative to WHERE clause, and is used **to match primary and foreign keys**.

An INNER join will only return rows from each table that have **matching rows** in the other.

2. Cross Join

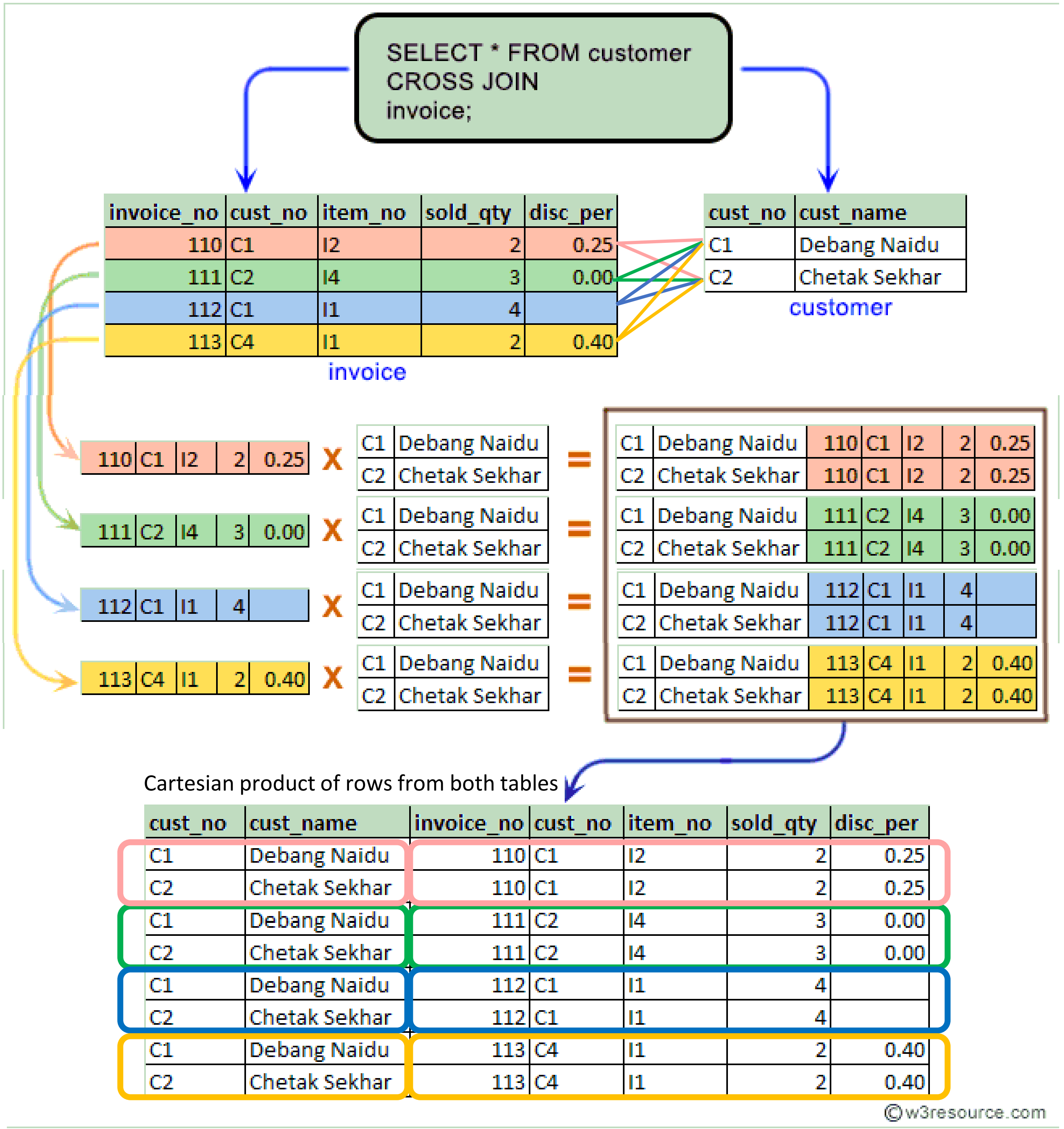


Cross JOIN

The **CROSS JOIN** joined **every row from the first table** with **every row from the second table.**

In other words, the cross join returns a Cartesian product of rows from both tables.

Image Reference: https://www.w3resource.com/PostgreSQL/postgresql-cross-join.php



```
SELECT * FROM customer
CROSS JOIN
invoice;
```

invoice_no	cust_no	item_no	sold_qty	disc_per
110	C1	I2	2	0.25
111	C2	I4	3	0.00
112	C1	I1	4	
113	C4	I1	2	0.40

invoice

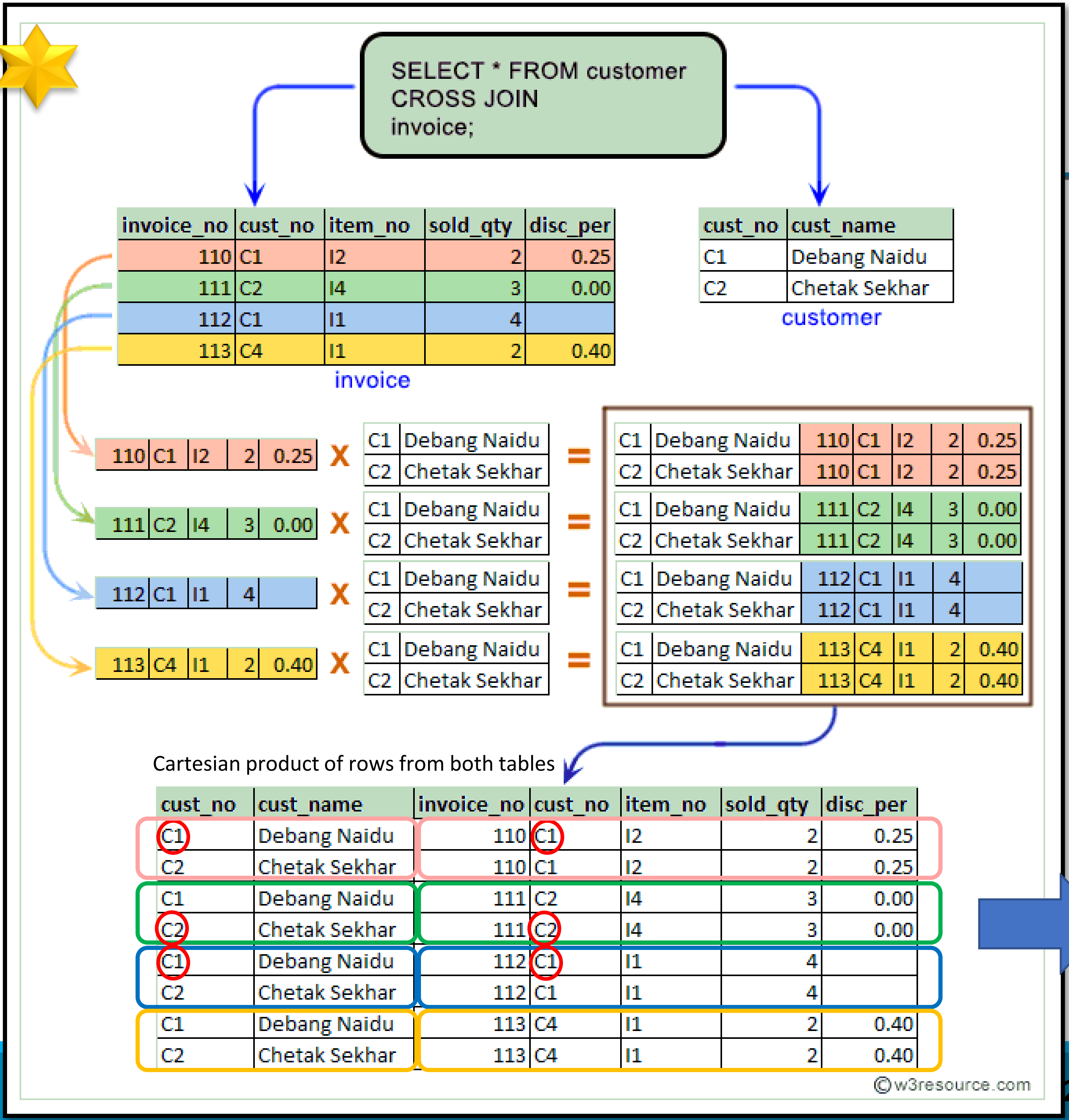
cust_no	cust_name
C1	Debang Naidu
C2	Chetak Sekhar

customer

C1	Debang Naidu	110	C1	I2	2	0.25
C2	Chetak Sekhar	110	C1	I2	2	0.25
C1	Debang Naidu	111	C2	I4	3	0.00
C2	Chetak Sekhar	111	C2	I4	3	0.00
C1	Debang Naidu	112	C1	I1	4	
C2	Chetak Sekhar	112	C1	I1	4	
C1	Debang Naidu	113	C4	I1	2	0.40
C2	Chetak Sekhar	113	C4	I1	2	0.40

Cartesian product of rows from both tables

cust_no	cust_name	invoice_no	cust_no	item_no	sold_qty	disc_per
C1	Debang Naidu	110	C1	I2	2	0.25
C2	Chetak Sekhar	110	C1	I2	2	0.25
C1	Debang Naidu	111	C2	I4	3	0.00
C2	Chetak Sekhar	111	C2	I4	3	0.00
C1	Debang Naidu	112	C1	I1	4	
C2	Chetak Sekhar	112	C1	I1	4	
C1	Debang Naidu	113	C4	I1	2	0.40
C2	Chetak Sekhar	113	C4	I1	2	0.40



```
SELECT * FROM customer
CROSS JOIN
invoice;
```

invoice_no	cust_no	item_no	sold_qty	disc_per
110	C1	I2	2	0.25
111	C2	I4	3	0.00
112	C1	I1	4	
113	C4	I1	2	0.40

invoice

cust_no	cust_name
C1	Debang Naidu
C2	Chetak Sekhar

customer

Cartesian product of rows from both tables

cust_no	cust_name	invoice_no	cust_no	item_no	sold_qty	disc_per
C1	Debang Naidu	110	C1	I2	2	0.25
C2	Chetak Sekhar	110	C1	I2	2	0.25
C1	Debang Naidu	111	C2	I4	3	0.00
C2	Chetak Sekhar	111	C2	I4	3	0.00
C1	Debang Naidu	112	C1	I1	4	
C2	Chetak Sekhar	112	C1	I1	4	
C1	Debang Naidu	113	C4	I1	2	0.40
C2	Chetak Sekhar	113	C4	I1	2	0.40

2. Cross Join

Select *
from customer Cross Join Invoice;

Select *
from customer Cross Join Invoice
Where customer.cust_no= Invoice. cust_no

Join table

cust_no	cust_name	invoice_no	cust_no	item_no	sold_qty	disc_per
C1	Debang Naidu	110	C1	I2	2	0.25
C2	Chetak Sekhar	111	C2	I4	3	0.00
C1	Debang Naidu	112	C1	I1	4	

Image Reference: https://www.w3resource.com/PostgreSQL/postgresql-cross-join.php

2. Cross Join

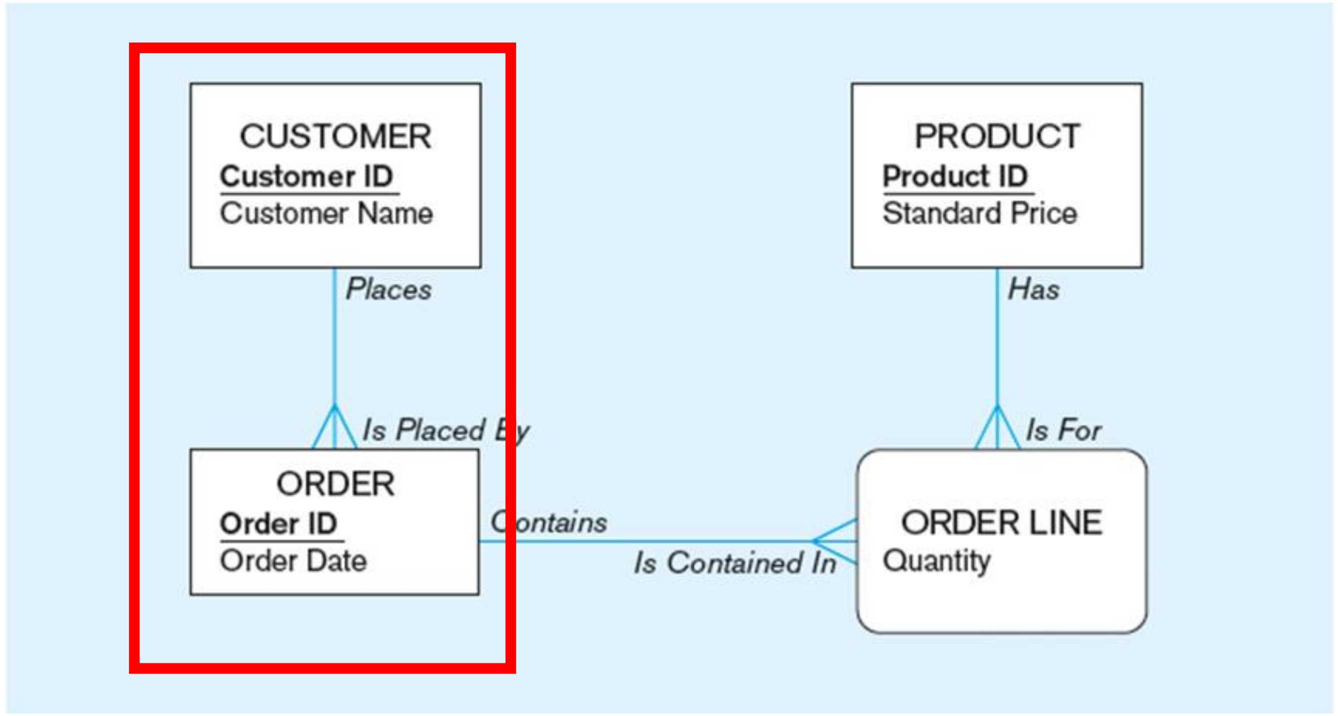
```
select Table-A.culumn1, culumn2, culumn3
from Table-A inner join Table-B
on Table-A. culumn1 = Table-B. culumn1;
```

```
select Table-A.culumn1, culumn2, culumn3
from Table-A Cross Join Table-B
where Table-A. culumn1 = Table-B. culumn1;
```

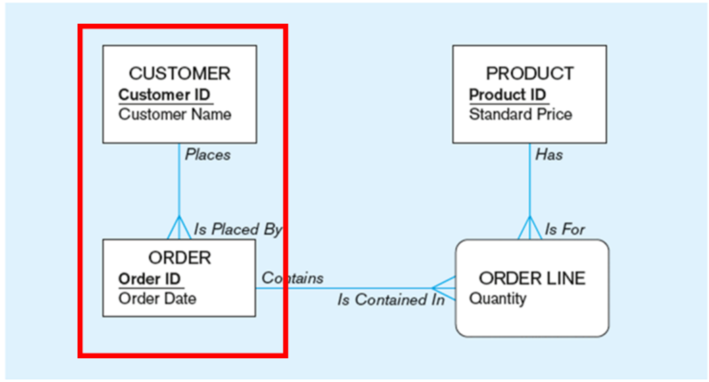
```
select Table-A.culumn1, culumn2, culumn3
from Table-A , Table-B
where Table-A. culumn1 = Table-B. culumn1;
```

Class Activity 8.2: Cross join

- Using cross product, determine the customer's **name** and **order number** for each customer who placed an order.



Example 2: Cross Join



Question: For each customer who placed an order, what is the customer's name and order number using cross product?

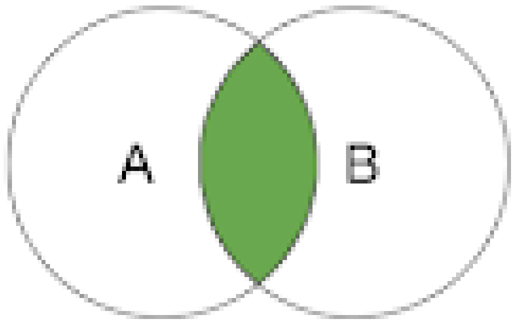
```
select orderid, order_t.customerid, customer_t.customerid, customername
from customer_t, order_t
where customer_t.customerid=order_t.customerid;

orderid | customerid | customerid |     customername
---------+------------+------------+---------------------
      1 |          4 |          4         | Eastern Furniture
      2 |          3 |          3         | Home Furnishings
      3 |          1 |          1         | Contemporary Casuals
      4 |          6 |          6         | Furniture Gallery
      5 |          4 |          4        | Eastern Furniture
      6 |          4 |          4         | Eastern Furniture
      7 |          1 |          1         | Contemporary Casuals
      8 |          4 |          4         | Eastern Furniture
      9 |          6 |          6         | Furniture Gallery
     19 |          4 |          4         | Eastern Furniture
     20 |          4 |          4         | Eastern Furniture
```

Notice places where a **dot** appears ...

a table name, followed by "." followed by the name of a column in that table.

This **disambiguates** which column we mean.

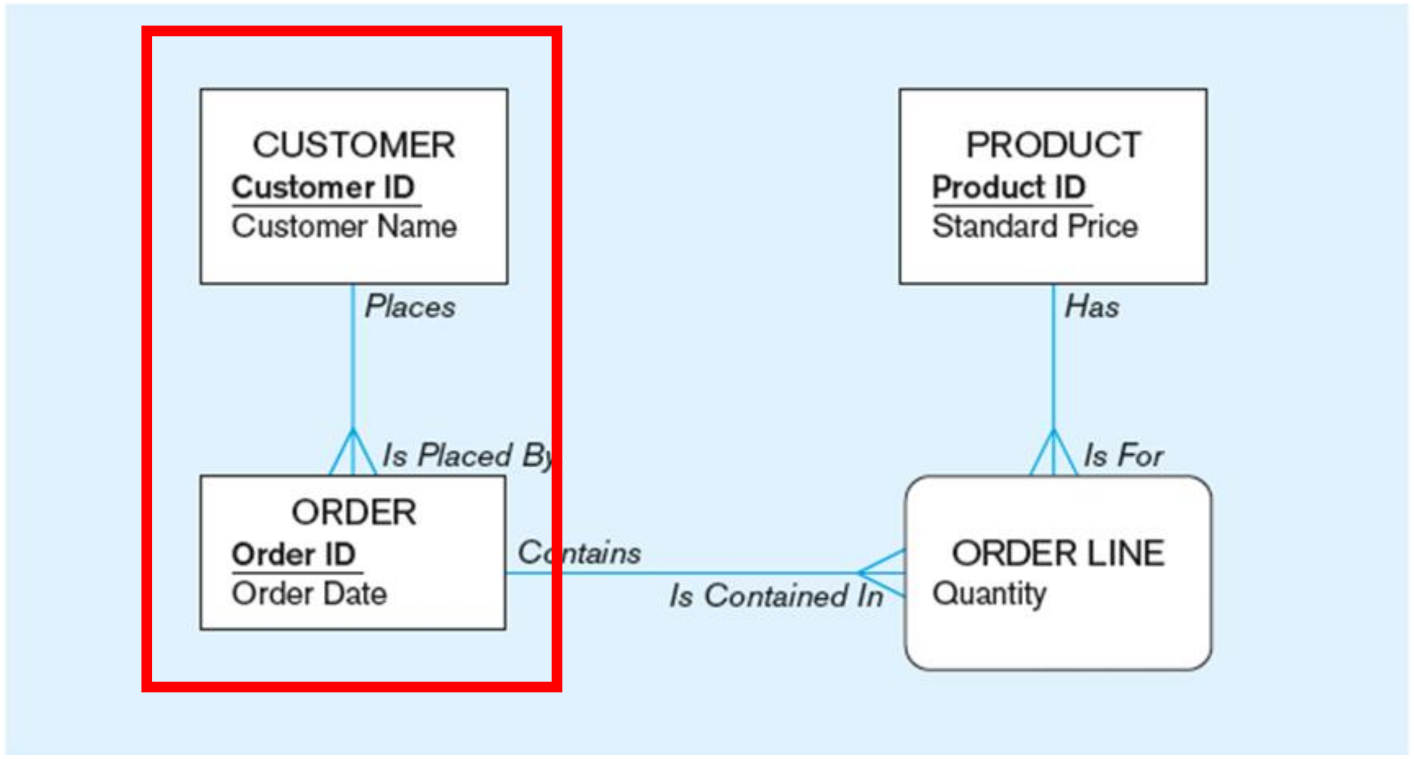
Inner Join Examples



INNER JOIN

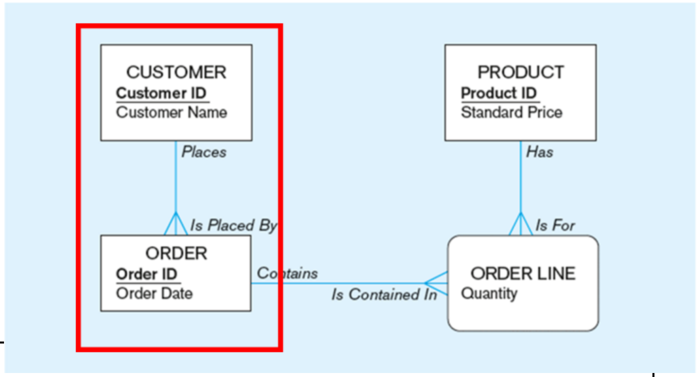
Class Activity 8.3: Inner join

- What are **order IDs, order dates** and **customer id** for **customer "Eastern Furniture"**?



Example 3: Inner join

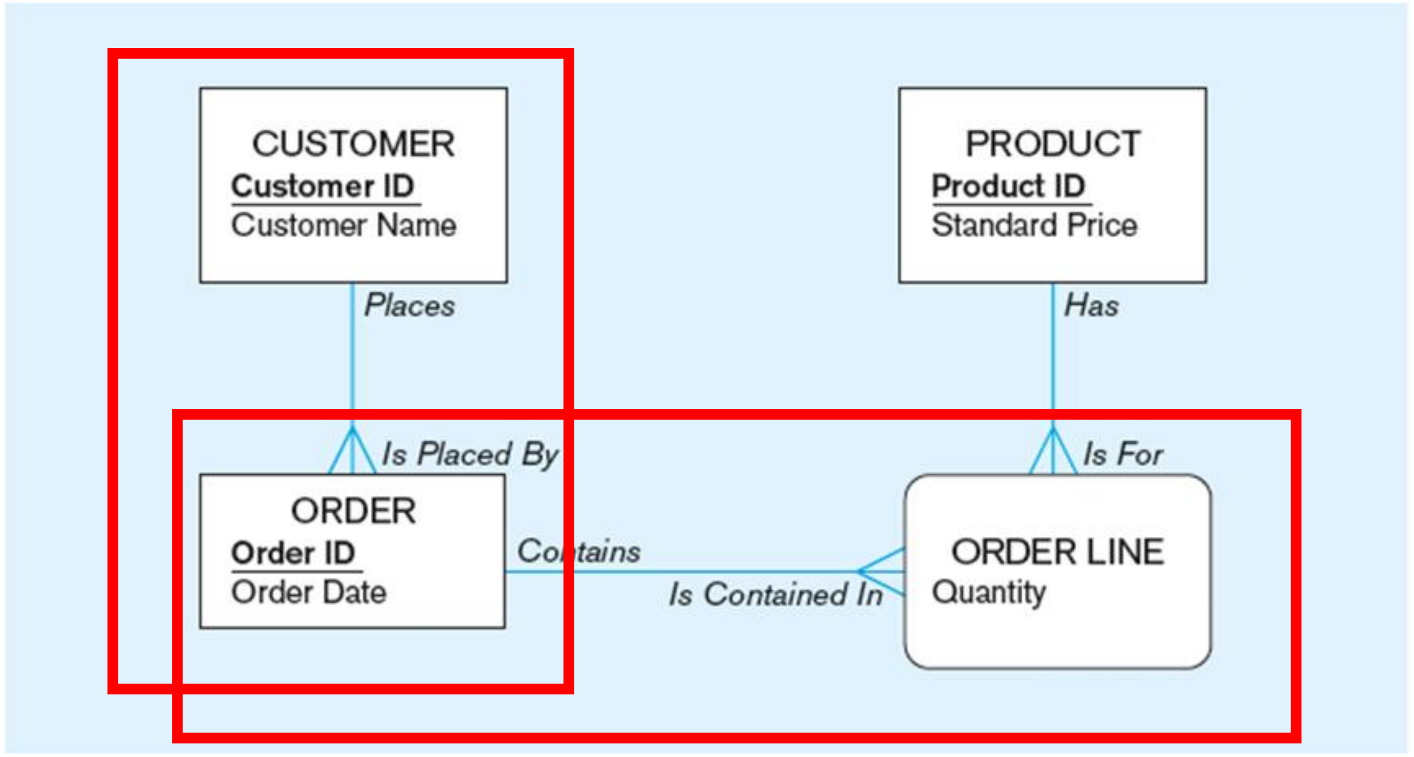
Question: What are order ids, order dates and customer id for customer “Eastern Furniture”?



```
select  order_t.customerid, customername,orderid, orderdate

from customer_t  inner join order_t on customer_t.customerid = order_t.customerid

where customername='Eastern Furniture';


 customerid |   customername    | orderid | orderdate
------------+-------------------+---------+------------
          4 | Eastern Furniture |       1 | 2009-09-08
          4 | Eastern Furniture |       5 | 2009-07-28
          4 | Eastern Furniture |       6 | 2009-08-27
          4 | Eastern Furniture |       8 | 2009-09-16
          4 | Eastern Furniture |      19 | 2010-03-05
          4 | Eastern Furniture |      20 | 2010-03-06
          4 | Eastern Furniture |      21 | 2010-03-06
          4 | Eastern Furniture |      22 | 2010-03-06
          4 | Eastern Furniture |      23 | 2010-03-06
                      .
                      .                  .
          4 | Eastern Furniture |      76 | 2010-09-15
(28 rows)
```

Class Activity 8.4: Multiple Tables join

What are order IDs, order dates, product id and customer id for customer “Eastern Furniture”?



Example 4: Multiple Table join

- What are order ids, order dates, product id and customer id for customer “Eastern Furniture”?

```
select customer_t .customerid, customername, order_t.orderid, orderdate, productid
from customer_t inner join  order_t on customer_t.customerid = order_t.customerid
                     inner join orderline_t on order_t. orderid = orderline_t. orderid
where customername='Eastern Furniture';
```

```
 customerid |   customername    | orderid | orderdate  | productid
------------+-------------------+---------+------------+-----------
          4 | Eastern Furniture |       1 | 2009-09-08 |         2
          4 | Eastern Furniture |       1 | 2009-09-08 |         6
          4 | Eastern Furniture |       1 | 2009-09-08 |        10
          4 | Eastern Furniture |       5 | 2009-07-28 |         1
          4 | Eastern Furniture |       5 | 2009-07-28 |         6
          4 | Eastern Furniture |      25 | 2010-03-10 |         2
          4 | Eastern Furniture |      26 | 2010-03-10 |         1
          4 | Eastern Furniture |      28 | 2010-03-10 |         1
          4 | Eastern Furniture |      39 | 2010-03-11 |         2
          4 | Eastern Furniture |      49 | 2010-03-11 |         1
          4 | Eastern Furniture |      63 | 2010-03-11 |         3
          4 | Eastern Furniture |      65 | 2010-03-11 |         4
          4 | Eastern Furniture |      69 | 2010-03-11 |         7
          4 | Eastern Furniture |      71 | 2010-09-08 |         3
(14 rows)
```

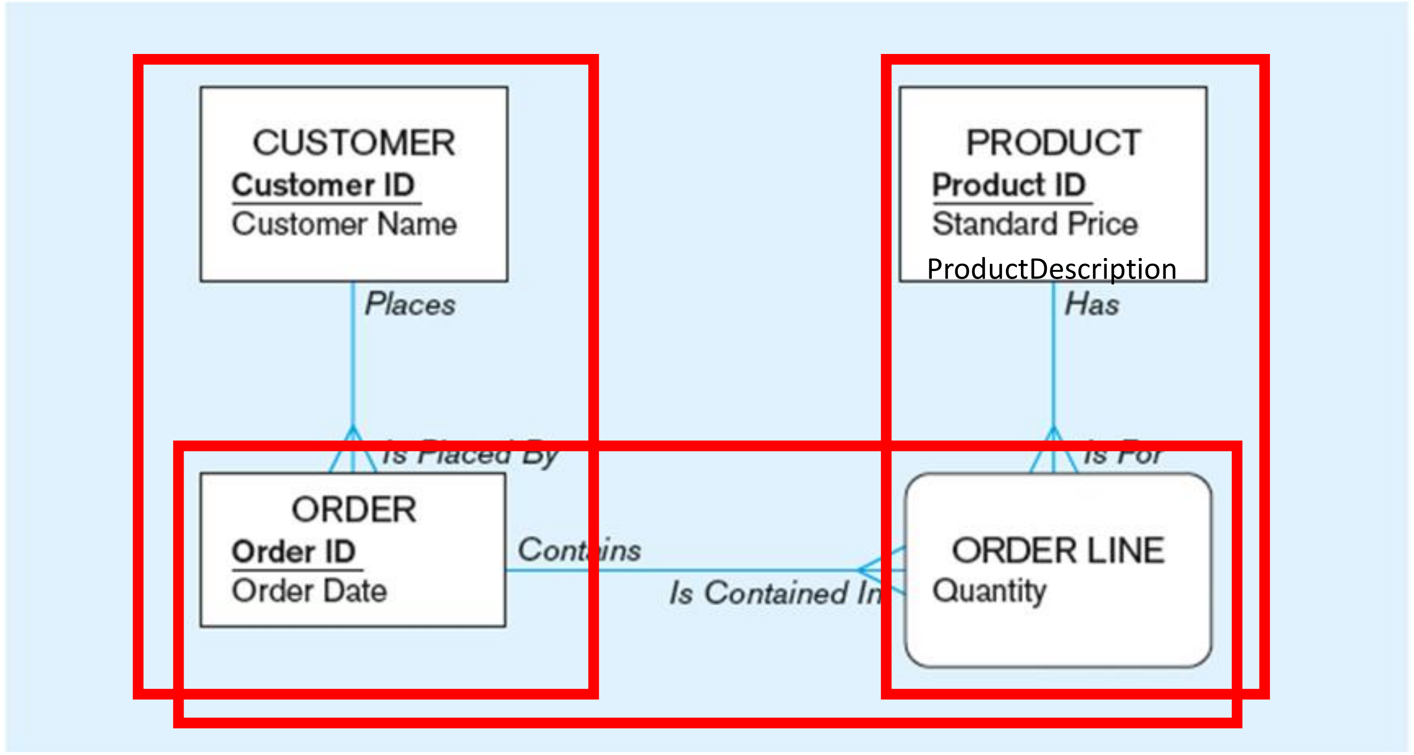
customer_t .customerid
Or
order_t .customerid

order_t.orderid
Or
orderLine_t.orderid

Class Activity 8.5: Multiple Tables join

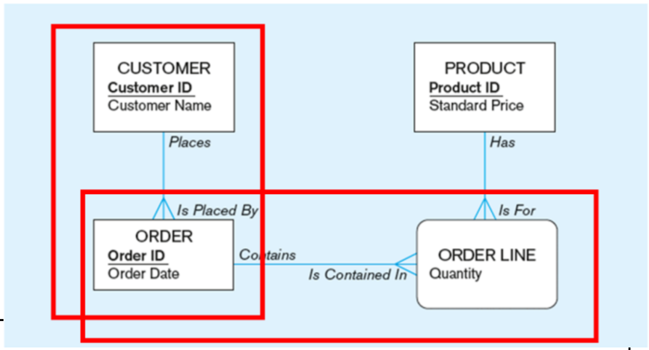
(from Chapter 1, Figure 1-3)

What are order IDs, order dates, product ID, product description and customer id for customer “Eastern Furniture”?



Example 5: Multiple Table join

Question: What are order ids, order dates, product id, product description and customer id for customer “Eastern Furniture”?



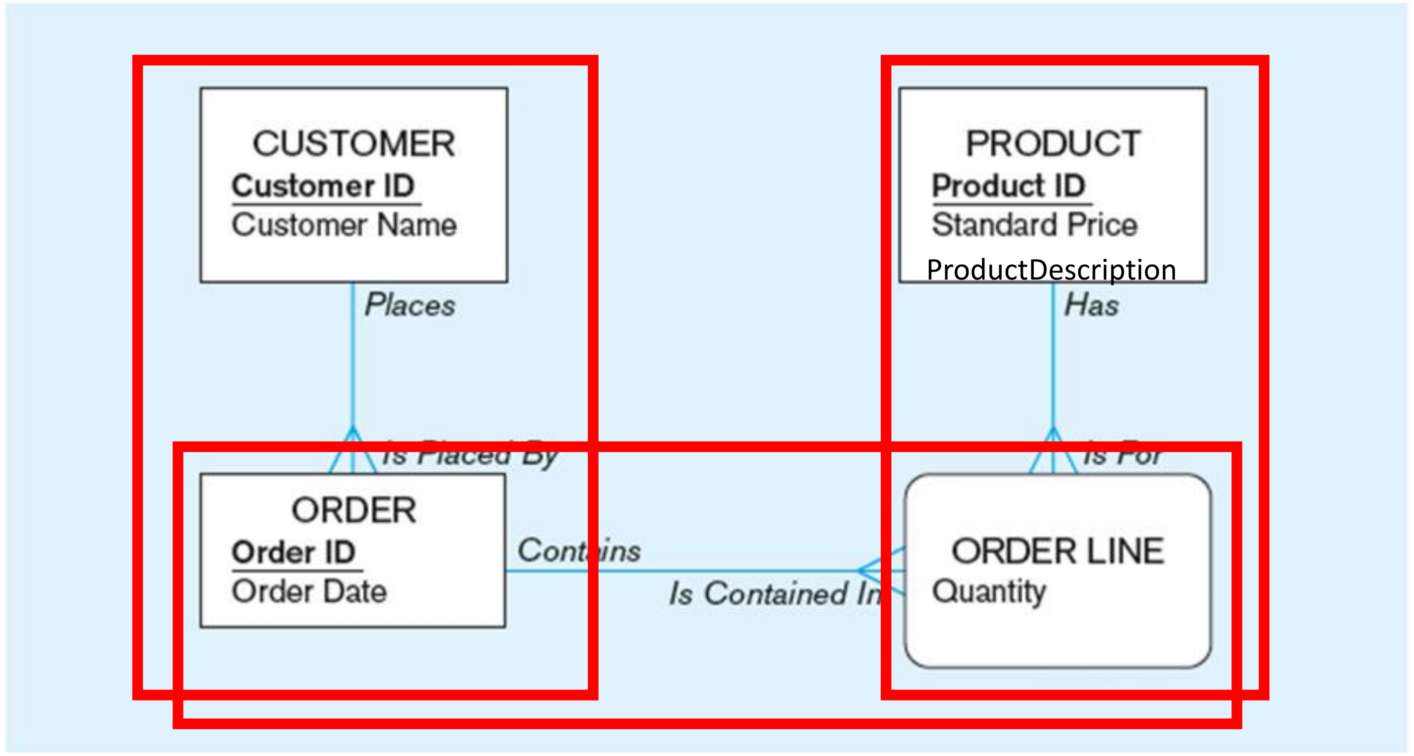
```
select  order_t.customerid, customername, order_t.orderid, orderdate, product_t.productid, productdescription
from customer_t inner join  order_t on customer_t.customerid = order_t.customerid
                inner join orderline_t on order_t.orderid = orderline_t.orderid
                inner join product_t on orderline_t.productid = product_t.productid
where customername='Eastern Furniture';
```

```
 customerid |   customername    | orderid | orderdate  | productid |  productdescription
------------+-------------------+---------+------------+-----------+---------------------
          4 | Eastern Furniture |       1 | 2009-09-08 |         2 | Birch Coffee Tables
          4 | Eastern Furniture |       1 | 2009-09-08 |         6 | 8-Drawer Dresser
          4 | Eastern Furniture |       1 | 2009-09-08 |        10 | 96 Bookcase
          4 | Eastern Furniture |       5 | 2009-07-28 |         1 | Cherry End Table
          4 | Eastern Furniture |       5 | 2009-07-28 |         6 | 8-Drawer Dresser
          4 | Eastern Furniture |      25 | 2010-03-10 |         2 | Birch Coffee Tables
          4 | Eastern Furniture |      26 | 2010-03-10 |         1 | Cherry End Table
          4 | Eastern Furniture |      28 | 2010-03-10 |         1 | Cherry End Table
          4 | Eastern Furniture |      39 | 2010-03-11 |         2 | Birch Coffee Tables
          4 | Eastern Furniture |      49 | 2010-03-11 |         1 | Cherry End Table
          4 | Eastern Furniture |      63 | 2010-03-11 |         3 | Oak Computer Desk
          4 | Eastern Furniture |      65 | 2010-03-11 |         4 | Entertainment Center
          4 | Eastern Furniture |      69 | 2010-03-11 |         7 | 48 Bookcase
          4 | Eastern Furniture |      71 | 2010-09-08 |         3 | Oak Computer Desk
(14 rows)
```

Class Activity 8.6: Multiple Tables join

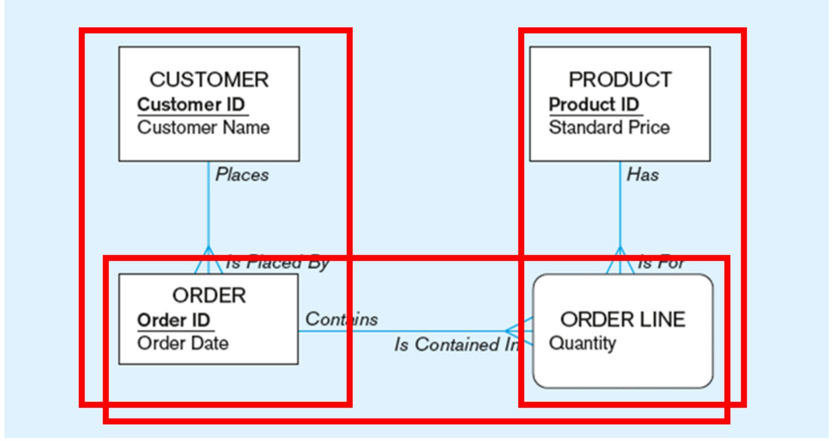
Assemble all information necessary to create an invoice for order number 4.

Invoice includes: customer id and name, order id, product description, order line id, and total price (ProductStandardPrice*OrderQuantity)



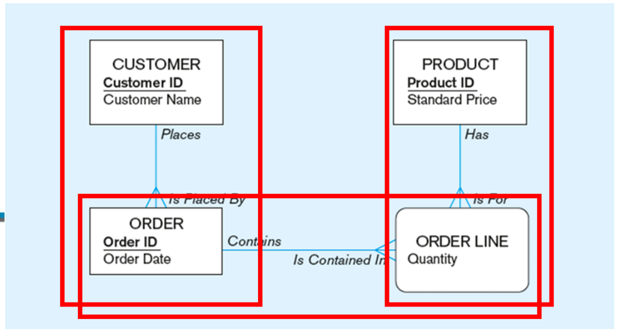
Example 6: Multiple Table join (using inner join)

Question: Assemble all information necessary to create an invoice for order number 1006.



```
select customer_t.customerid, customername, order_t.orderid, product_t.productdescription,
        (productstandardprice* orderedquantity) as price
from customer_t inner join  order_t on customer_t.customerid = order_t.customerid
                            inner join orderline_t on order_t.orderid = orderline_t.orderid
                            inner join product _t on orderline_t.productid = product _t.productid
where order_t.orderid=4;


 customerid |   customername    | orderid |  productdescription      |  price
------------+-------------------+---------+--------------------------+------------
          6 | Furniture Gallery |       4 | Oak Computer Desk        |  750.00
          6 | Furniture Gallery |       4 | Entertainment Center     |   0.00
          6 | Furniture Gallery |       4 | Writer's Desk            |  975.00
          6 | Furniture Gallery |       4 | 8-Drawer Dresser         |   2250.00
(4 rows)
```

Example 6: Multiple Table Join Example (using Cross join).



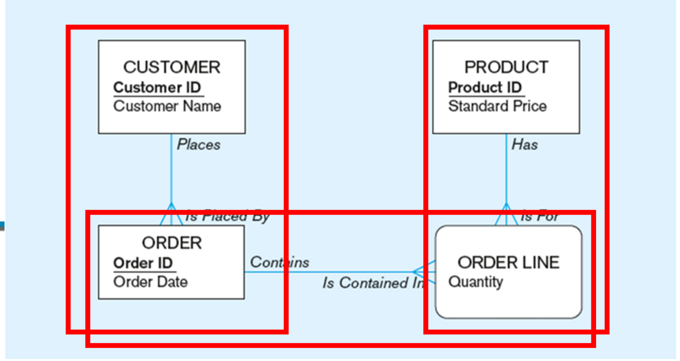
Question: Assemble all information necessary to create an invoice for order number 1006.

```
SELECT Customer_T.CustomerID, CustomerName, CustomerAddress,
   CustomerCity, CustomerState, CustomerPostalCode, Order_T.OrderID,
   OrderDate, OrderedQuantity, ProductDescription, StandardPrice,
   (OrderedQuantity * ProductStandardPrice)
FROM Customer_T, Order_T, OrderLine_T, Product_T
   WHERE Order_T.CustomerID = Customer_T.CustomerID
      AND Order_T.OrderID = OrderLine_T.OrderID
      AND OrderLine_T.ProductID = Product_T.ProductID
      AND Order_T.OrderID = 1006;
```

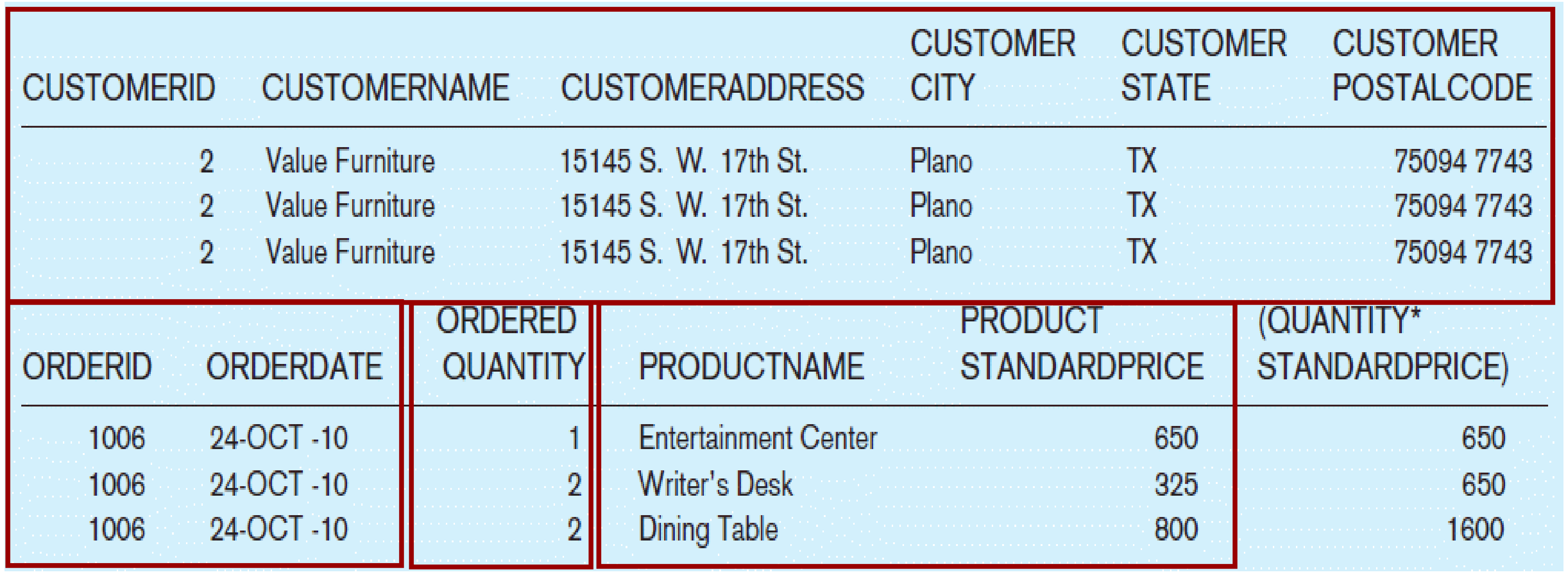
Four tables involved in this join

Each pair of tables requires an equality-check condition in the WHERE clause, matching primary keys against foreign keys.

Example 6: Results from a four-table join (Figure 7-4)



From CUSTOMER_T table



CUSTOMERID	CUSTOMERNAME	CUSTOMERADDRESS	CUSTOMER CITY	CUSTOMER STATE	CUSTOMER POSTALCODE
2	Value Furniture	15145 S. W. 17th St.	Plano	TX	75094 7743
2	Value Furniture	15145 S. W. 17th St.	Plano	TX	75094 7743
2	Value Furniture	15145 S. W. 17th St.	Plano	TX	75094 7743

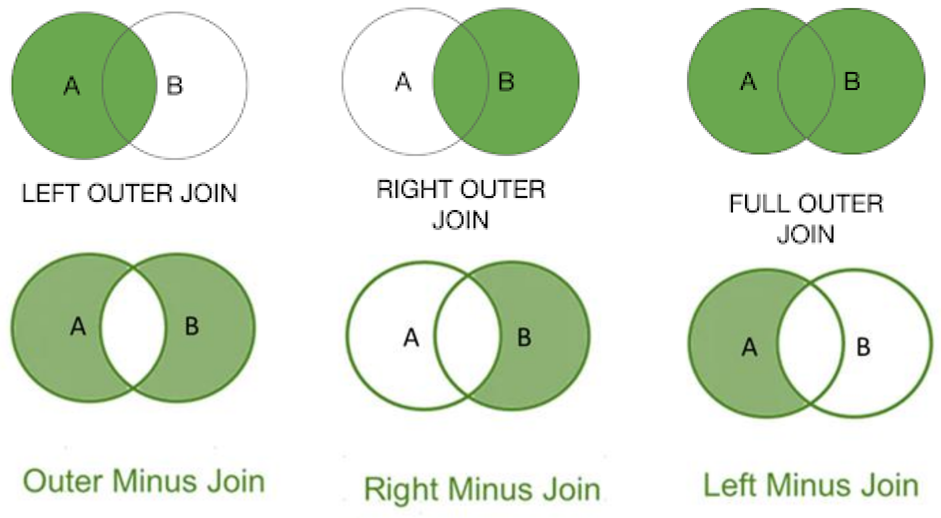
ORDERID	ORDERDATE	ORDERED QUANTITY	PRODUCTNAME	PRODUCT STANDARDPRICE	(QUANTITY* STANDARDPRICE)
1006	24-OCT -10	1	Entertainment Center	650	650
1006	24-OCT -10	2	Writer's Desk	325	650
1006	24-OCT -10	2	Dining Table	800	1600

From ORDER_T table

From ORDERLINE_T table

From PRODUCT_T table

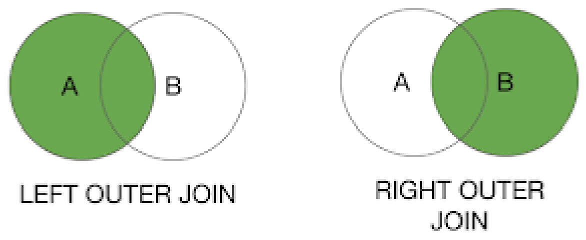
3. Outer joins



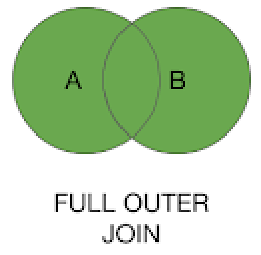
3. Outer join

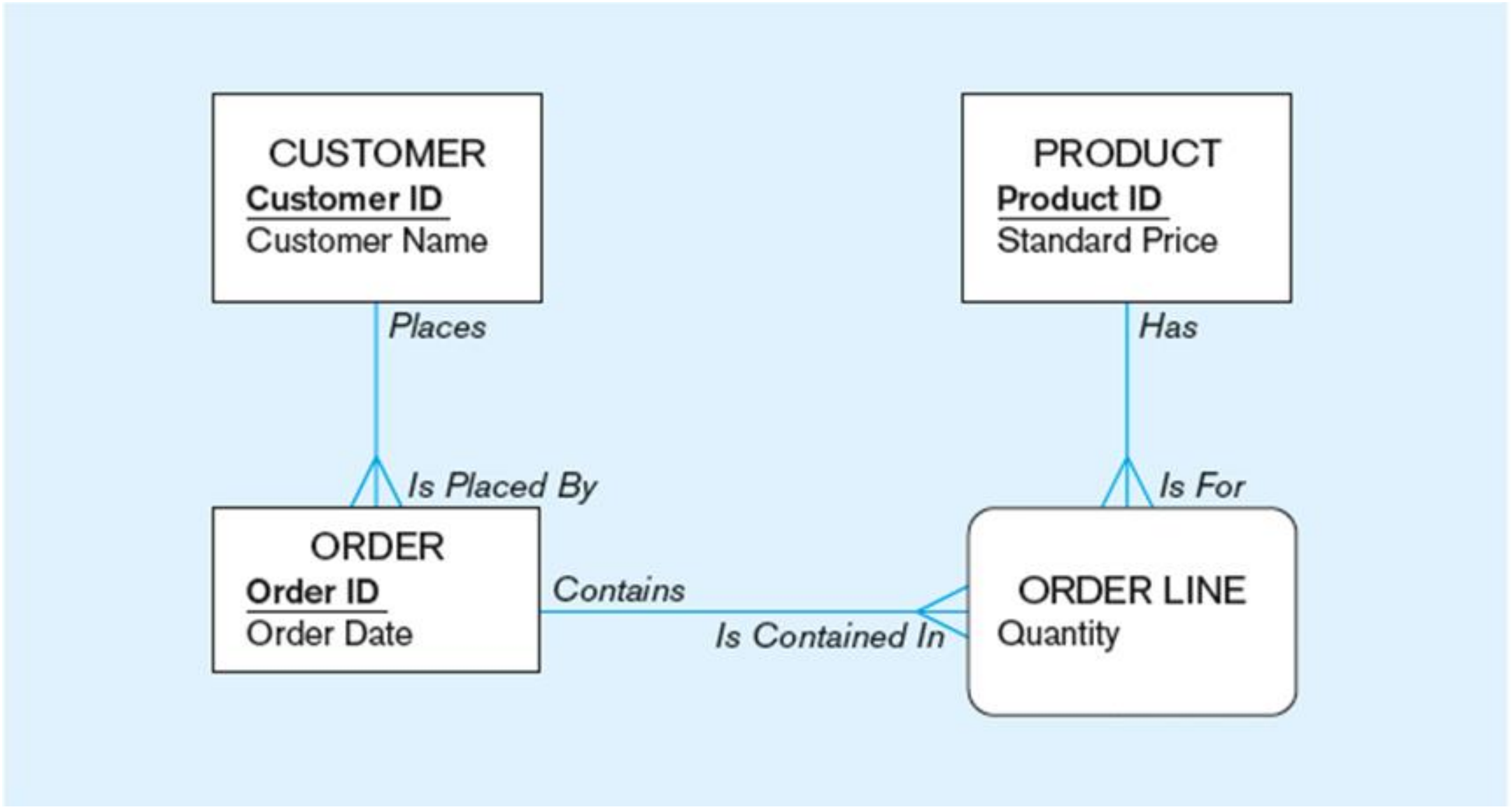
- **Outer join:** a join in which rows that do not have matching values in common columns are nonetheless included in the result table (as opposed to *inner* join, in which rows must have matching values in order to appear in the result table)

- **Outer join Types:**
  - **Left/right outer join**



  - **Full outer join**




CUSTOMER
Customer ID
Customer Name
Places
Is Placed By
PRODUCT
Product ID
Standard Price
Has
Is For
ORDER
Order ID
Order Date
Contains
Is Contained In
ORDER LINE
Quantity

3. Left/right outer join

Left outer join produces a **complete set of records from Table A**, with the matching records (where available) in Table B. If there is no match, the right side will contain null.

Table A

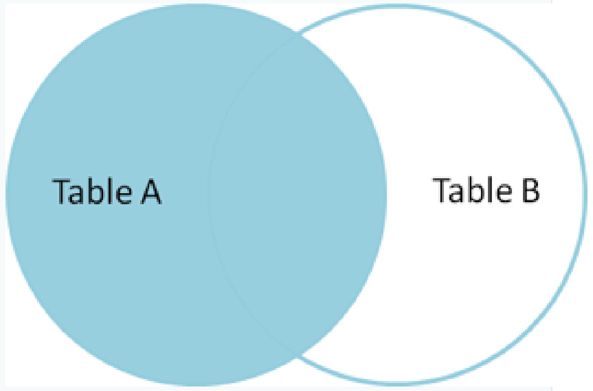
id	name
1	Pirate
2	Monkey
3	Ninja
4	Spaghetti

Table B

code	name
1	Rutabaga
2	Pirate
3	Darth Vader
4	Ninja

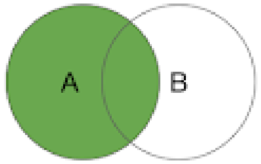
```
SELECT *
FROM TableA LEFT OUTER JOIN TableB
        ON TableA.name = TableB.name
```

id	name	code	name
1	Pirate	2	Pirate
2	Monkey	null	null
3	Ninja	4	Ninja
4	Spaghetti	null	null

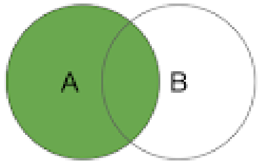


3. Left/right outer join

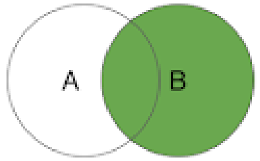
```
SELECT *
FROM TableA LEFT OUTER JOIN TableB
        ON TableA.name = TableB.name
```



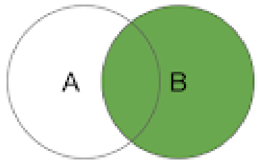
```
SELECT *
FROM TableB RIGHT OUTER JOIN TableA
        ON TableA.name = TableB.name
```



```
SELECT *
FROM TableA RIGHT OUTER JOIN TableB
        ON TableA.name = TableB.name
```



```
SELECT *
FROM TableB LEFT OUTER JOIN TableA
        ON TableA.name = TableB.name
```



Class Activity 8.7: Outer join

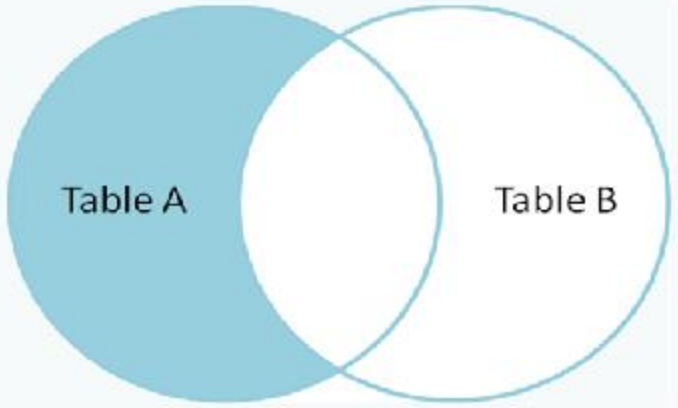
- Produce the set of records only in Table A, but not in Table B.

Note: You can change this query to show the required rows

```
Table A                  Table B
id name                code   name
-- ----                 --   ----
1  Pirate               1    Rutabaga
2  Monkey               2    Pirate
3  Ninja                3    Darth Vader
4  Spaghetti            4    Ninja
```

```
SELECT * FROM TableA
LEFT OUTER JOIN TableB
ON TableA.name = TableB.name
```

```
id    name      code    name
--    ----      --      ----
1     Pirate    2       Pirate
2     Monkey    null    null
3     Ninja     4       Ninja
4     Spaghetti null    null
```

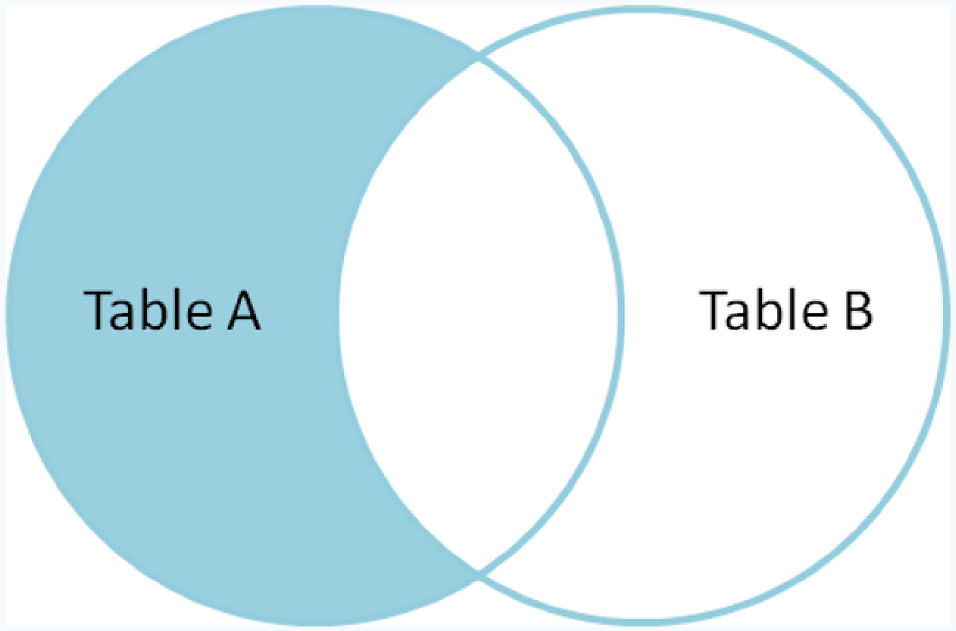


Solution to the Class Activity 8.7:

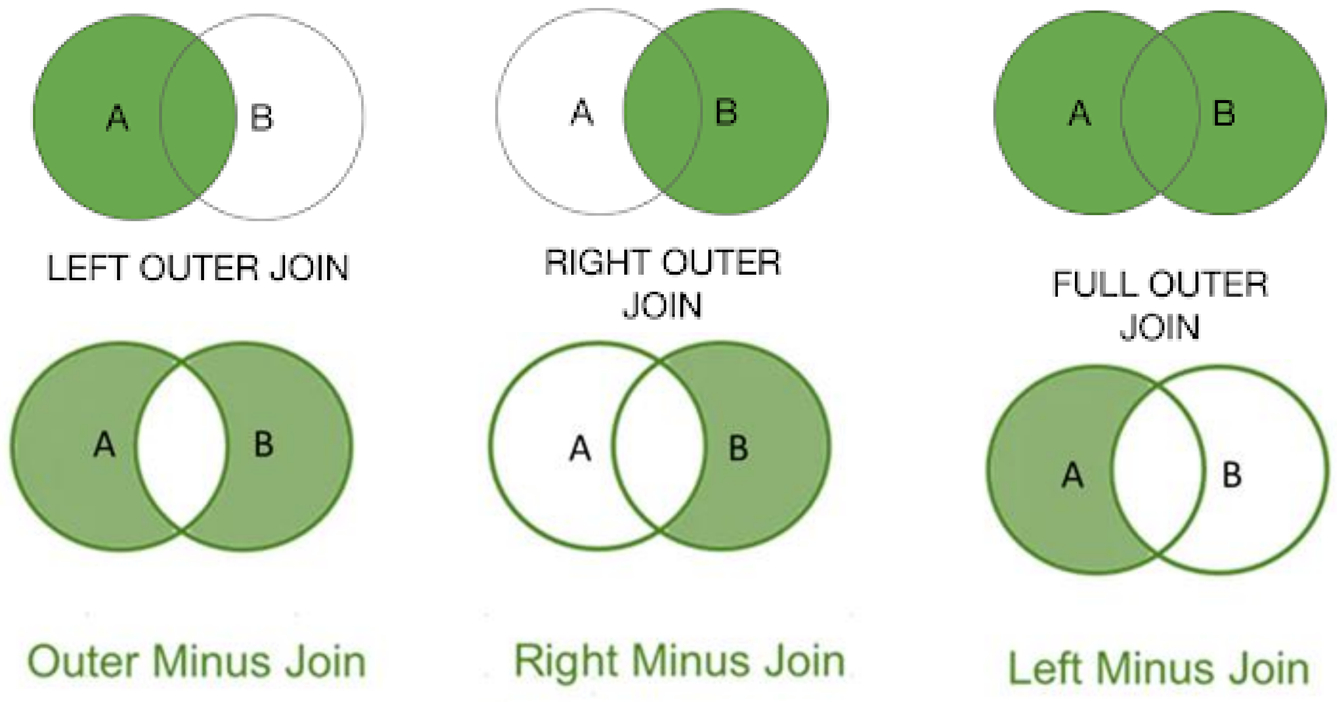
➢ **Produce the set of records only in Table A, but not in Table B.**

To answer this question, perform the same left outer join, then **exclude the records we don't want from the right side via a where clause.**

```
SELECT * FROM TableA
LEFT OUTER JOIN TableB
ON TableA.name = TableB.name
WHERE TableB.code  is null

id   name        id     name
--   ----        --     ----
2    Monkey      null   null
4    Spaghetti   null   null
```



Outer Join Examples

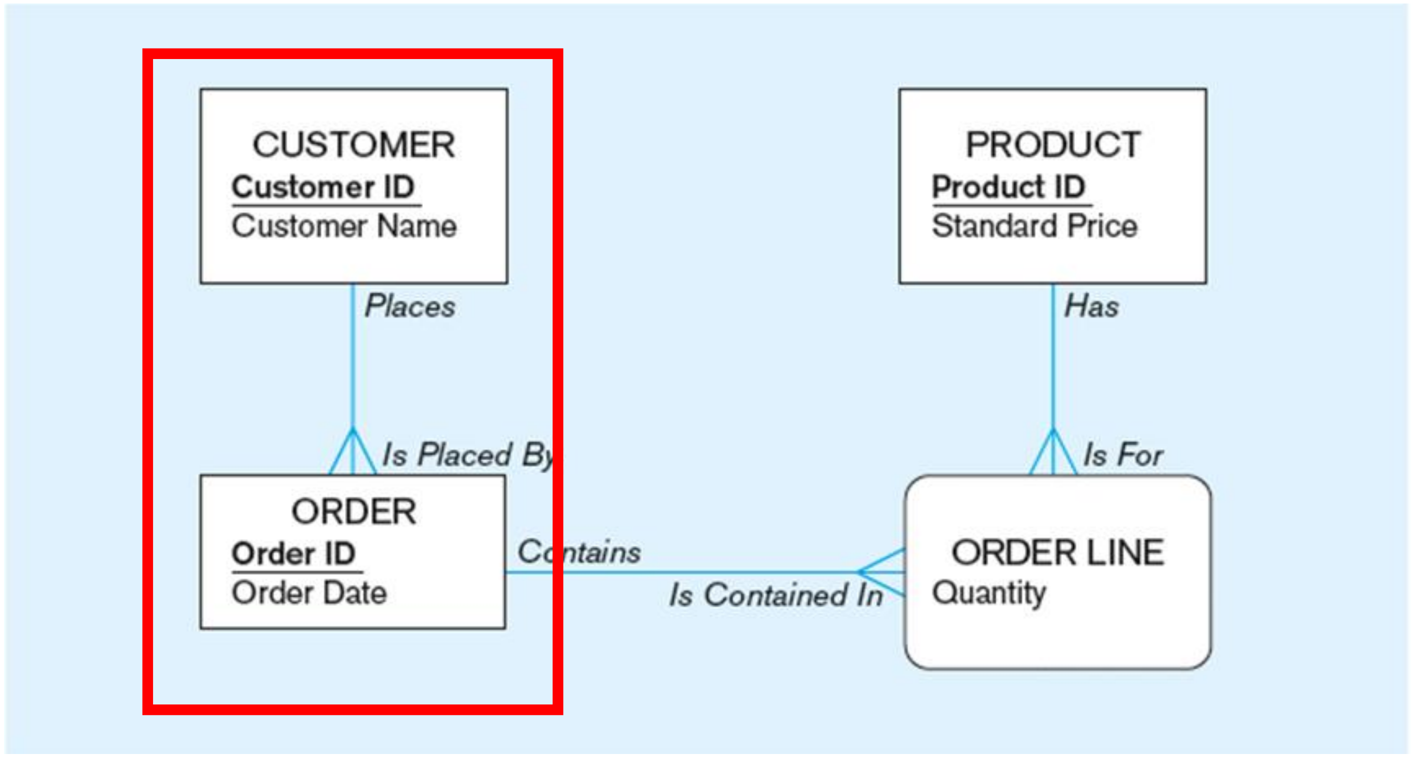


Visualization of different join types with results returned in shaded area

Class Activity 8.8: Outer join

List the customer name, ID number, and order number for all customers.

Include customer information even for customers that do not have an order.



Example 8: Outer Join Example

- List the customer name, ID number, and order number for all customers. Include customer information even for customers that do not have an order.

```
select customer_t.customerid , customername, orderid
from customer_t left outer join order_t
on customer_t.customerid=order_t.customerid;
```

LEFT OUTER JOIN clause causes customer data to appear even if there is no corresponding order data

Unlike INNER join, this will include customer rows with no matching order rows

Left Outer Join Results

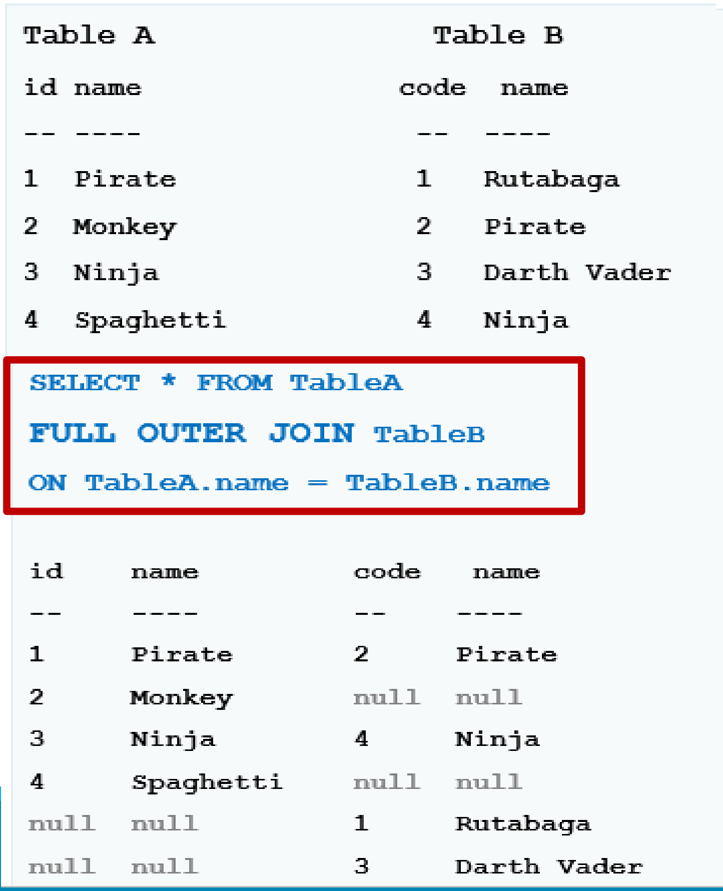
```
select customer_t.customerid , customername, orderid
from customer_t left outer join order_t
on customer_t.customerid=order_t.customerid;
```

Unlike INNER join, this will include customer rows with no matching order rows

```
customerid |    customername     | orderid
-----------+---------------------+---------
         4 | Eastern Furniture   |       1
         3 | Home Furnishings    |       2
         1 | Contemporary Casuals |      3
         6 | Furniture Gallery   |       4
         4 | Eastern Furniture   |       5
         4 | Eastern Furniture   |       6
         1 | Contemporary Casuals |      7
         4 | Eastern Furniture   |       8
         6 | Furniture Gallery   |       9
         4 | Eastern Furniture   |      19
         4 | Eastern Furniture   |      20
         4 | Eastern Furniture   |      21
         4 | Eastern Furniture   |      22
         4 | Eastern Furniture   |      23
         1 | Contemporary Casuals |     24
         4 | Eastern Furniture   |      25
         4 | Eastern Furniture   |      26
         4 | Eastern Furniture   |      27
         4 | Eastern Furniture   |      28
         4 | Eastern Furniture   |      29
         4 | Eastern Furniture   |      30
        15 | Janet's Collection  |      31
        15 | Janet's Collection  |      32
        15 | Janet's Collection  |      34
                  .
                  .
                  .
         4 | Eastern Furniture   |      71
        12 | Flanigan Furniture  |      73
         1 | Contemporary Casuals |     75
         4 | Eastern Furniture   |      76
         2 | Value Furnitures    |    null
         5 | Impressions         |    null
         7 | New Furniture       |    null
(61 rows)
```

3. Type of Join: Full outer join

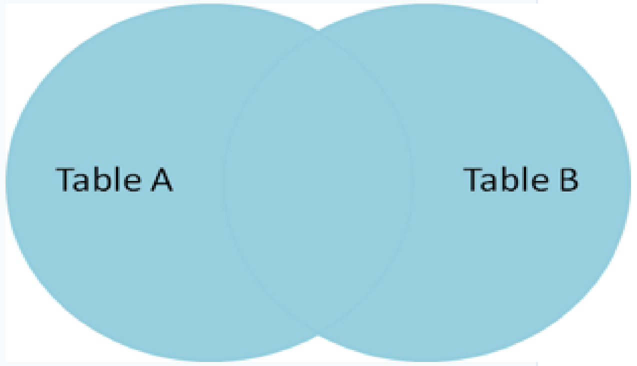
Full outer join: includes all columns from each table , and an instance for each row of each table with matching records from both sides where available. If there is no match, the missing side will contain null.



```
Table A                    Table B
id name                  code  name
-- ----                    --  ----
1  Pirate                  1   Rutabaga
2  Monkey                  2   Pirate
3  Ninja                   3   Darth Vader
4  Spaghetti               4   Ninja
```

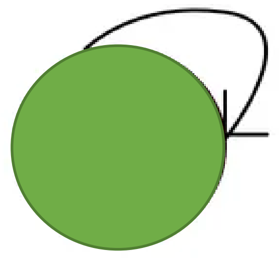
```sql
SELECT * FROM TableA
FULL OUTER JOIN TableB
ON TableA.name = TableB.name
```

```
id     name          code    name
--     ----          --      ----
1      Pirate        2       Pirate
2      Monkey        null    null
3      Ninja         4       Ninja
4      Spaghetti     null    null
null   null          1       Rutabaga
null   null          3       Darth Vader
```



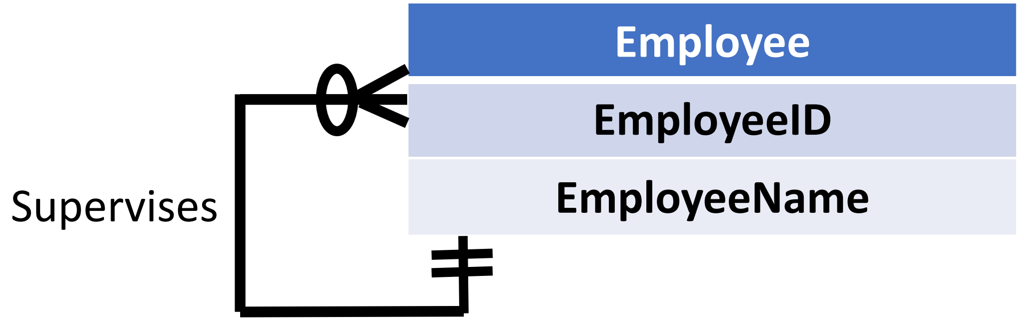
4. Self-Join

(Unary relationship)

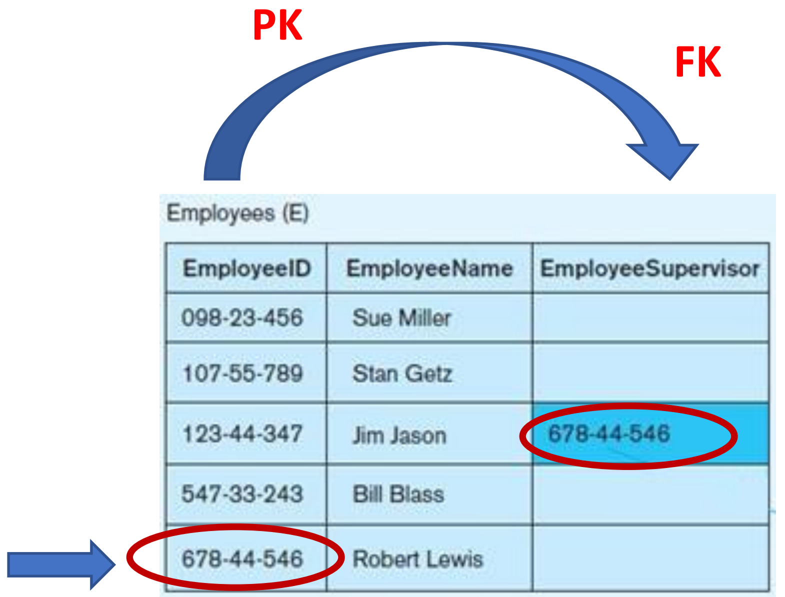

Self Join

Class Activity 8.9: Self join

- What are the employee ID and name of each employee and the name of his/her supervisor (label the supervisor’s name Manager)



- What is the name of supervisor of **Jim Jason**?



Employees (E)

EmployeeID	EmployeeName	EmployeeSupervisor
098-23-456	Sue Miller	
107-55-789	Stan Getz	
123-44-347	Jim Jason	678-44-546
547-33-243	Bill Blass	
678-44-546	Robert Lewis	

```
select * from EMPLOYEE E, EMPLOYEE M;
```

E employeeid	E employeename	E employeesupervisor	M employeeid	M employeename	M employeesupervisor
107-55-789	Stan Getz		107-55-789	Stan Getz	
107-55-789	Stan Getz		123-44-347	Jim Jason	678-44-546
107-55-789	Stan Getz		547-33-243	Bill Blass	
107-55-789	Stan Getz		678-44-546	Robert Lewis	
107-55-789	Stan Getz		098-23-456	Sue Miller	547-33-243
123-44-347	Jim Jason	678-44-546	107-55-789	Stan Getz	
123-44-347	Jim Jason	678-44-546	123-44-347	Jim Jason	678-44-546
123-44-347	Jim Jason	678-44-546	547-33-243	Bill Blass	
123-44-347	Jim Jason	678-44-546	678-44-546	Robert Lewis	
123-44-347	Jim Jason	678-44-546	098-23-456	Sue Miller	547-33-243
547-33-243	Bill Blass		107-55-789	Stan Getz	
547-33-243	Bill Blass		123-44-347	Jim Jason	678-44-546
547-33-243	Bill Blass		547-33-243	Bill Blass	
547-33-243	Bill Blass		678-44-546	Robert Lewis	
547-33-243	Bill Blass		098-23-456	Sue Miller	547-33-243
678-44-546	Robert Lewis		107-55-789	Stan Getz	
678-44-546	Robert Lewis		123-44-347	Jim Jason	678-44-546
678-44-546	Robert Lewis		547-33-243	Bill Blass	
678-44-546	Robert Lewis		678-44-546	Robert Lewis	
678-44-546	Robert Lewis		098-23-456	Sue Miller	547-33-243
098-23-456	Sue Miller	547-33-243	107-55-789	Stan Getz	
098-23-456	Sue Miller	547-33-243	123-44-347	Jim Jason	678-44-546
098-23-456	Sue Miller	547-33-243	547-33-243	Bill Blass	
098-23-456	Sue Miller	547-33-243	678-44-546	Robert Lewis	
098-23-456	Sue Miller	547-33-243	098-23-456	Sue Miller	547-33-243

EMPLOYEE E

employeeid	employeename	employeesupervisor
107-55-789	Stan Getz	
123-44-347	Jim Jason	678-44-546
547-33-243	Bill Blass	
678-44-546	Robert Lewis	
098-23-456	Sue Miller	547-33-243

EMPLOYEE M

employeeid	employeename	employeesupervisor
107-55-789	Stan Getz	
123-44-347	Jim Jason	678-44-546
547-33-243	Bill Blass	
678-44-546	Robert Lewis	
098-23-456	Sue Miller	547-33-243

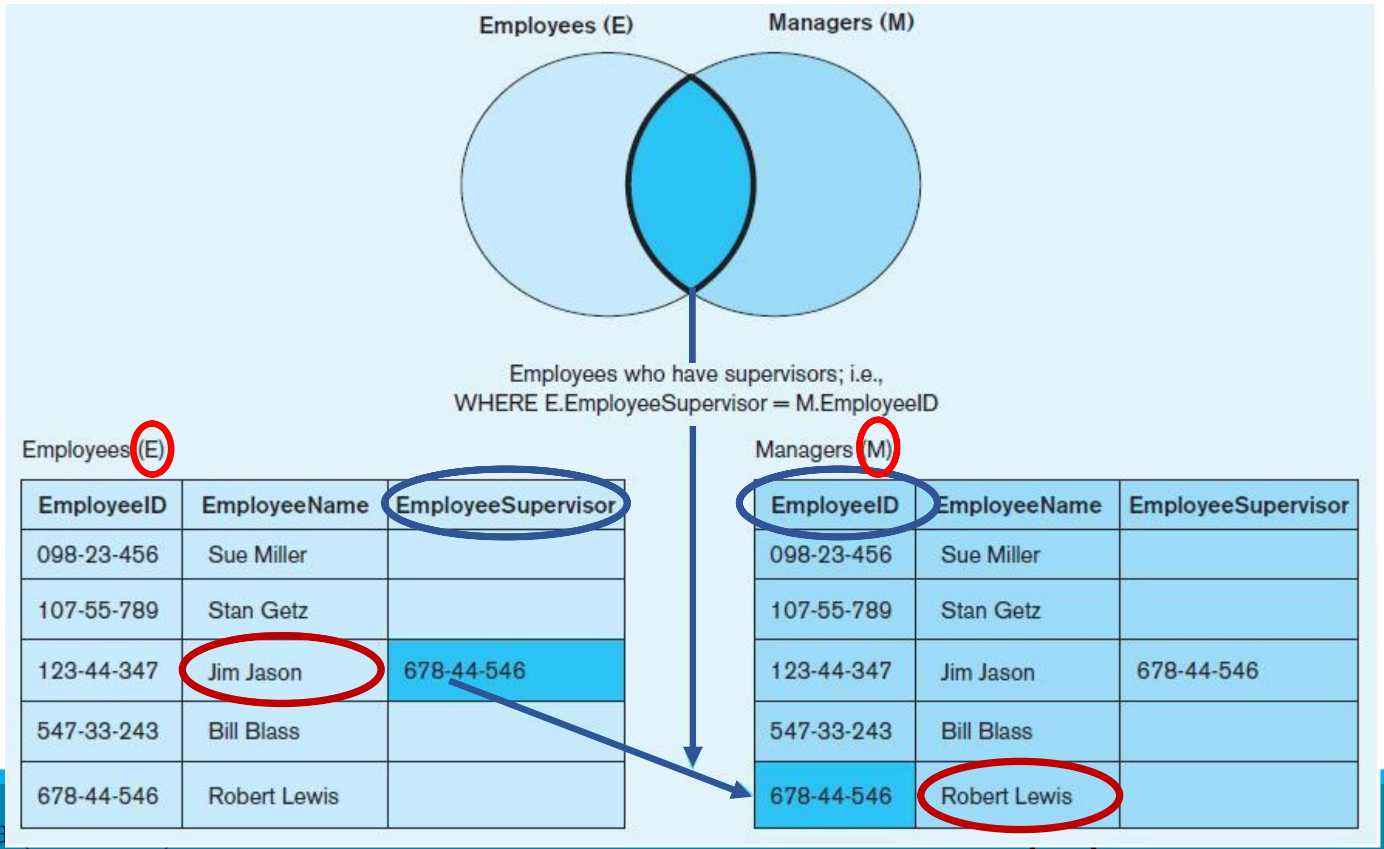
```
select E.EmployeeID, E.EmployeeName, M.EmployeeName as Manager
from EMPLOYEE E, EMPLOYEE M
where E.EmployeeSupervisor = M.EmployeeID;
```

E	E	M
employeeid	employeename	manager
123-44-347	Jim Jason	Robert Lewis
098-23-456	Sue Miller	Bill Blass

Example 9: Self-join (Figure 7-5)

```sql
SELECT E.EmployeeID, E.EmployeeName, M.EmployeeName AS Manager
  FROM Employee  E, Employee  M
  WHERE E.EmployeeSupervisor = M.EmployeeID;
```

Why **cross join**? Can we use **inner join** instead?
See Slide 56 …



Employees (E)
Managers (M)
Employees who have supervisors; i.e.,
WHERE E.EmployeeSupervisor = M.EmployeeID

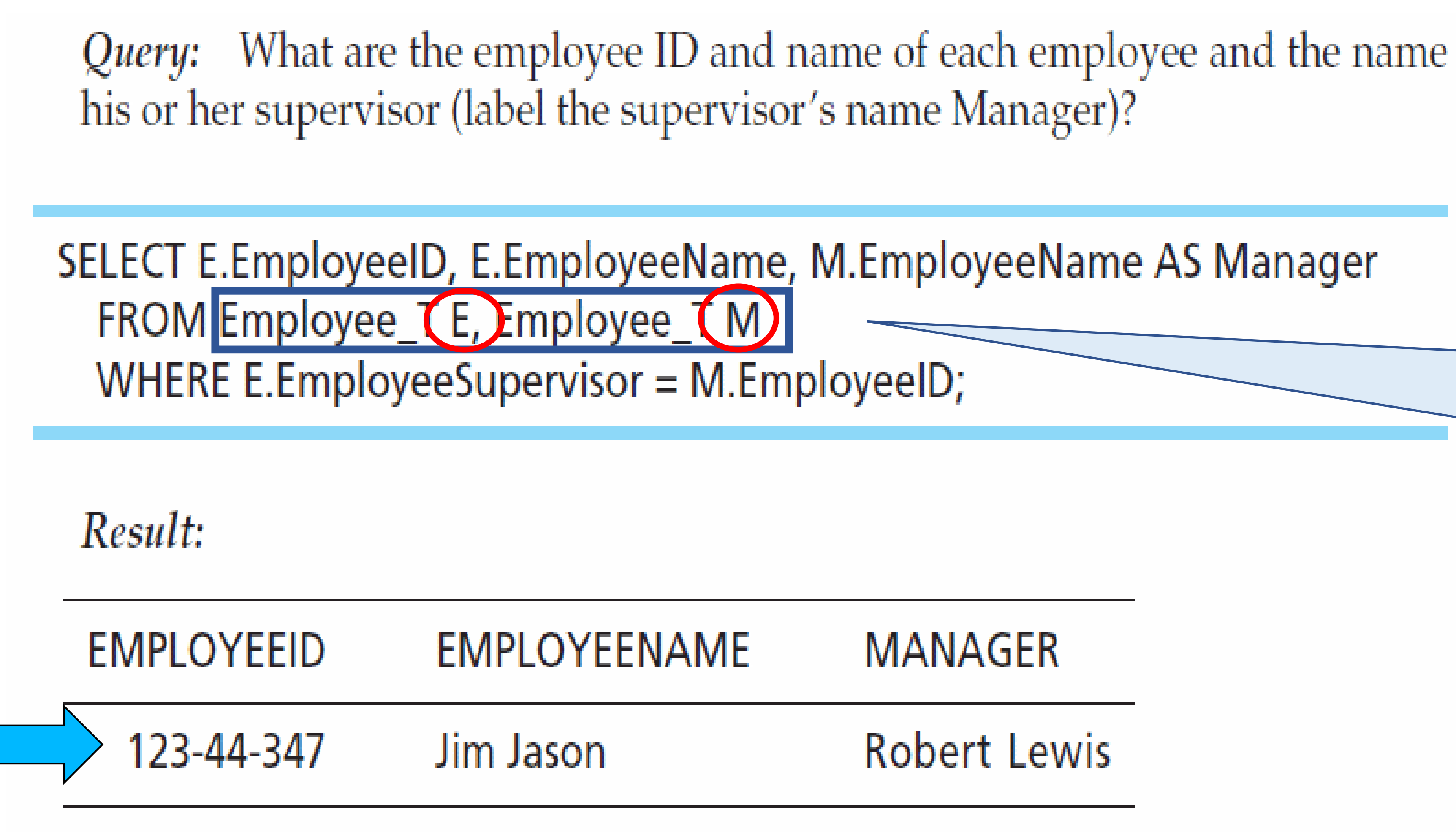
Employees (E)

EmployeeID	EmployeeName	EmployeeSupervisor
098-23-456	Sue Miller	
107-55-789	Stan Getz	
123-44-347	Jim Jason	678-44-546
547-33-243	Bill Blass	
678-44-546	Robert Lewis	

Managers (M)

EmployeeID	EmployeeName	EmployeeSupervisor
098-23-456	Sue Miller	
107-55-789	Stan Getz	
123-44-347	Jim Jason	678-44-546
547-33-243	Bill Blass	
678-44-546	Robert Lewis	

Example 9: Self-Join



Query: What are the employee ID and name of each employee and the name of his or her supervisor (label the supervisor's name Manager)?

```
SELECT E.EmployeeID, E.EmployeeName, M.EmployeeName AS Manager
  FROM Employee_T E, Employee_T M
  WHERE E.EmployeeSupervisor = M.EmployeeID;
```

Result:

EMPLOYEEID	EMPLOYEENAME	MANAGER
123-44-347	Jim Jason	Robert Lewis

The same table is used on both sides of the join; distinguished using table **aliases**

Self-joins are usually used on tables with **unary relationships**.

Can we answer this question now?

Question: I need the information about **my life** and **my success** after **COVID-19 gone.**

Question: I need the information about **my life** and **my success** after **COVID-19 gone.**

MyLife_T

HappinessID	HappinessName	HppinessStartDate	HppinessEndDate	COVID_19
1755	Pass DF	09/03/2020	null	Gone
1899	Graduated	09/03/2019	null	Came
…	…	…	…	…

MySuccess_T

SuccessID	SuccessName	SuccessDate	HappinessID
1967	Got HD Grade in PF	8/10/2019	1755
2055	Got HD Grade in DF	null	1755
3798	Start my job in NASA	null	1899
…	…	…	…

```
Select * from MyLife_T Inner Join MySuccess_T on MyLife_T.HappinessID = MySuccess_T. HappinessID
where COVID_19 = 'Gone';
```

HappinessID	HappinessName	HppinessStartDate	HppinessEndDate	COVID_19	SuccessID	SuccessName	SuccessDate	HappinessID
1755	Pass DF	09/03/2020	null	Gone	1967	Got HD Grade in PF	8/10/2019	1755
1755	Pass DF	09/03/2020	null	Gone	2055	Got HD Grade in DF	null	1755
…	…	…	…	…	…	…	…	…

Extra Information

1. Type of Join: Natural Join

- **Natural join**: an inner-join in which one of the duplicate columns is eliminated in the result table.

```
SELECT *
    FROM TableA INNER JOIN TableB
    ON TableA.name = TableB.name
```

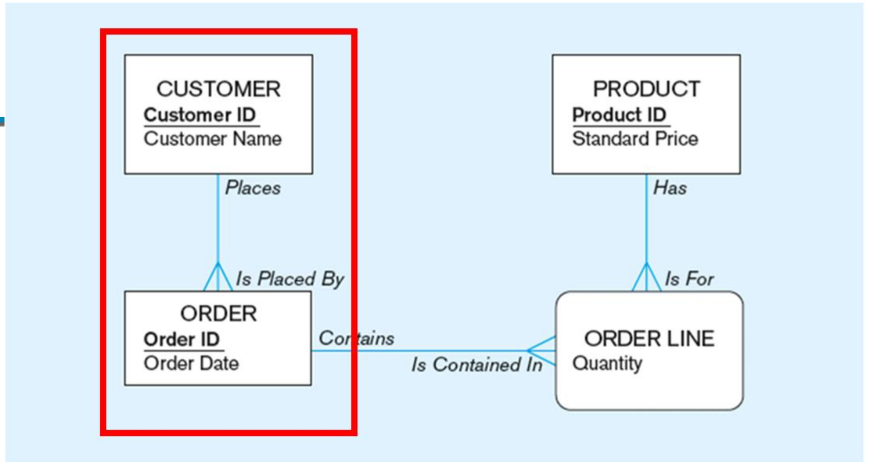
```
SELECT *
FROM TableA  NATURAL JOIN TableB
```

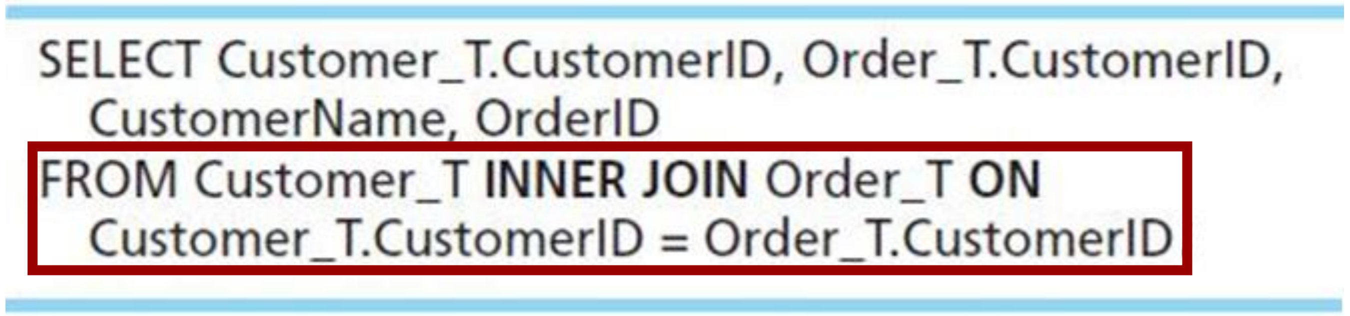
Note: Please don’t use natural join

Example 10: Natural Join

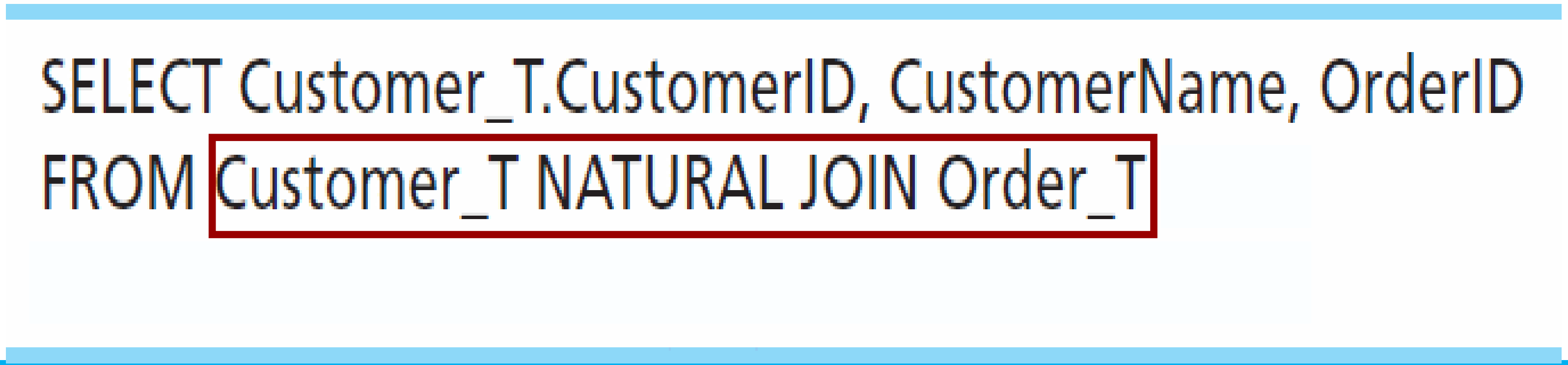
- For each customer who placed an order, what is the customer's name and order number?

  Answer this question using natural join





```
SELECT Customer_T.CustomerID, Order_T.CustomerID,
   CustomerName, OrderID
FROM Customer_T INNER JOIN Order_T ON
   Customer_T.CustomerID = Order_T.CustomerID
```



```
SELECT Customer_T.CustomerID, CustomerName, OrderID
FROM Customer_T NATURAL JOIN Order_T
```

2. Union

- **Union -**The results of two queries can be combined using **union**

query1 **UNION** [ALL] query2

UNION effectively appends the result of **query2** to the result of **query1**.

Furthermore, it eliminates duplicate rows from its result, in the same way as DISTINCT, unless UNION ALL is used.

2 Union

```
SELECT column_1, column_2 FROM Table_A
UNION [All]
SELECT column_1, column_2 FROM Table_B;
```

Rules:

- Both queries must return the same number of columns.
- The corresponding columns in the queries must have compatible data types.

Example 11: Union

Query 1

```
select customer_t.customerid from customer_t left join order_t
on customer_t.customerid=order_t.customerid where customer_t.customerid<15
union
select customer_t.customerid from customer_t right join order_t
on customer_t.customerid=order_t.customerid where customer_t.customerid<15;
```

```
customerid
--------------------------
          1
          2
          3
          4
          5
          6
          7
          8
          9
         12
         13
         14
(12 rows)
```

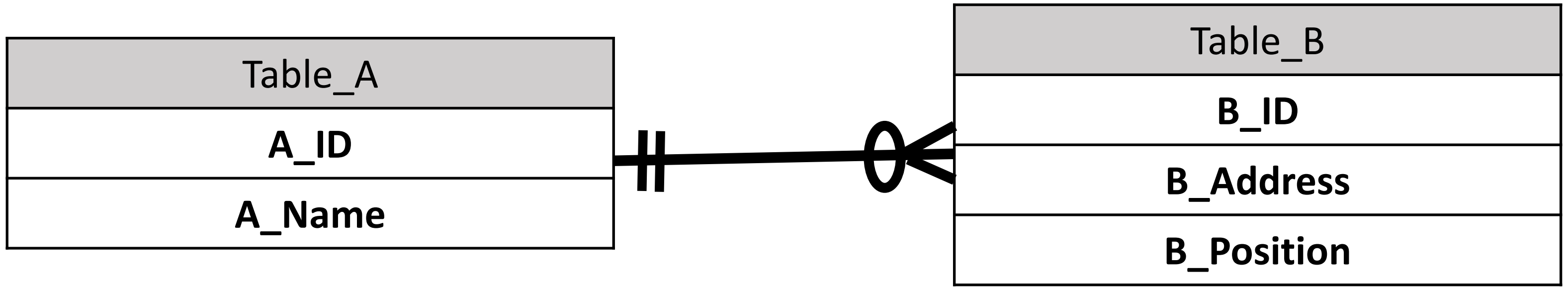
Query 2

```
select distinct (customer_t.customerid)
from customer_t full outer join order_t
on customer_t.customerid=order_t.customerid
where customer_t.customerid<15;
```

Compare the results of Query 1 and Query 2

Summary (Joins): Inner join, Cross join Self join

Note 1: you can form any type of joins (cross, inner, self) based on the equality of the PK and FK values in different tables or one table (self Join).



```
select *
from Table_A , Table_B
where Table_A.PK = Table_B.FK;
```

```
select *
from Table_A cross join Table_B
where Table_A.PK = Table_B.FK;
```

```
select *
from Table_A inner join Table_B
on Table_A.PK = Table_B.FK;
```

Note 2: In self join, we may also form the join over the equality of the values in different columns rather than PK and FK (See tutorial Question 5)

Summary

Joins

1. Inner join
2. Cross join
3. Outer join
4. Self join

Extra information

1. Natural join
2. Unions