UTS:ENGINEERING AND INFORMATION TECHNOLOGY

UTS UNIVERSITY OF TECHNOLOGY, SYDNEY

lecture 9: SQL III

Subquery

Main reference:

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

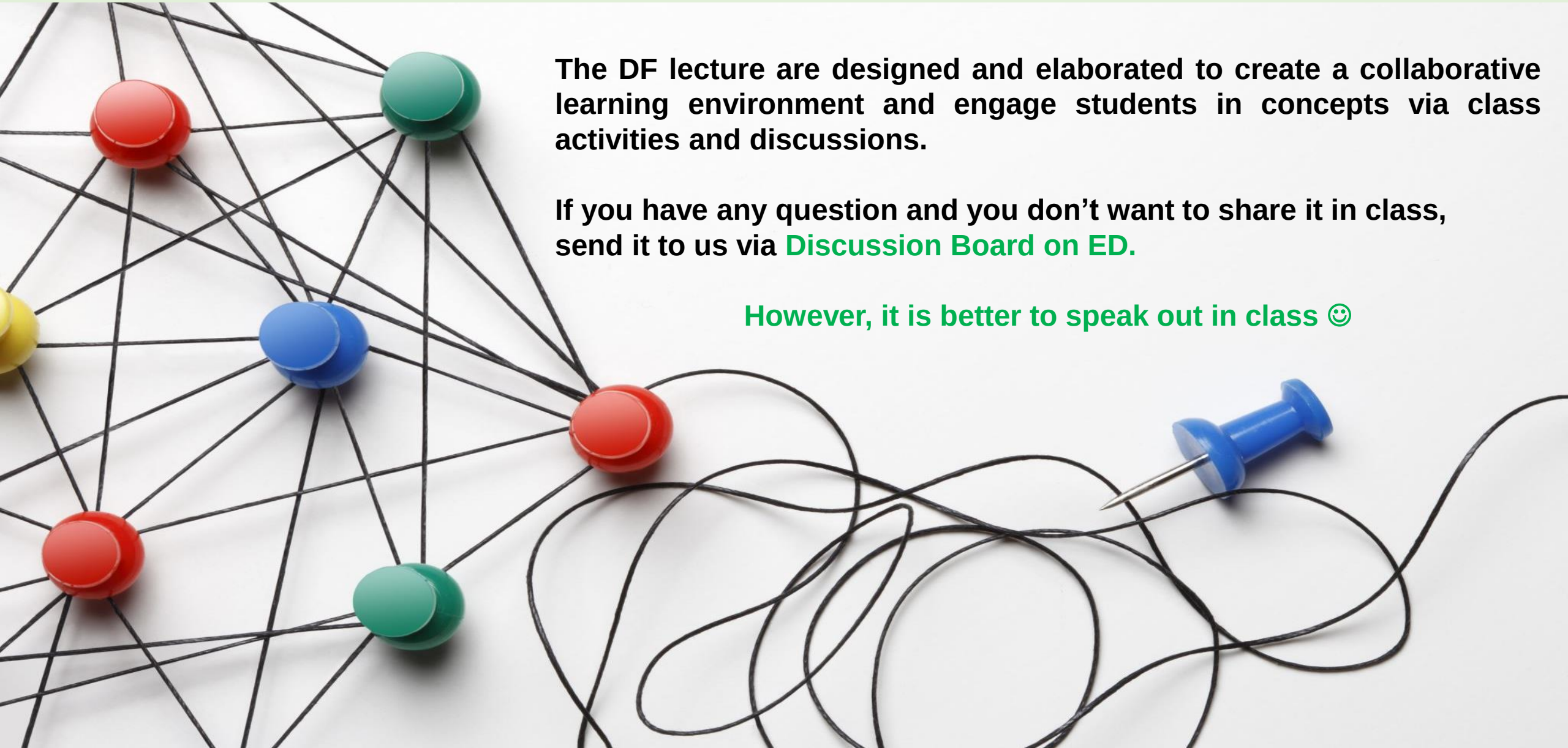
Subject Coordinator and Instructor:

Dr. Danna (Fahimeh) Ramezani

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

UTS CRICOS PROVIDER CODE: 00099F

Participations and Discussions

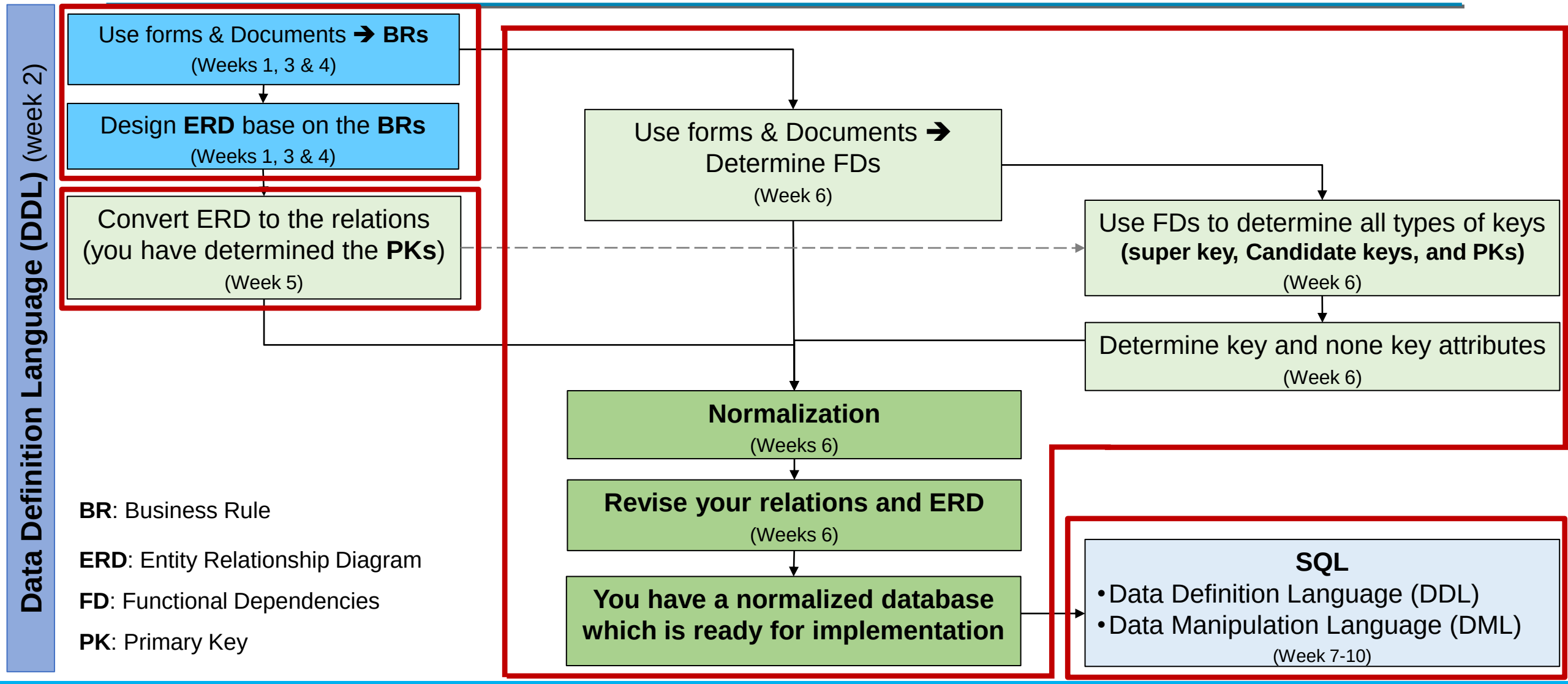


The DF lecture are designed and elaborated to create a collaborative learning environment and engage students in concepts via class activities and discussions.

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



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 success** after **COVID-19 is 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
…	…	…	…

Join

```
Select SuccessID, SuccessName, SuccessDate, MySuccess_T.HappinessID
from MyLife_T Inner Join MySuccess_T on MyLife_T.HappinessID = MySuccess_T. HappinessID
where COVID_19 = 'Gone';
```

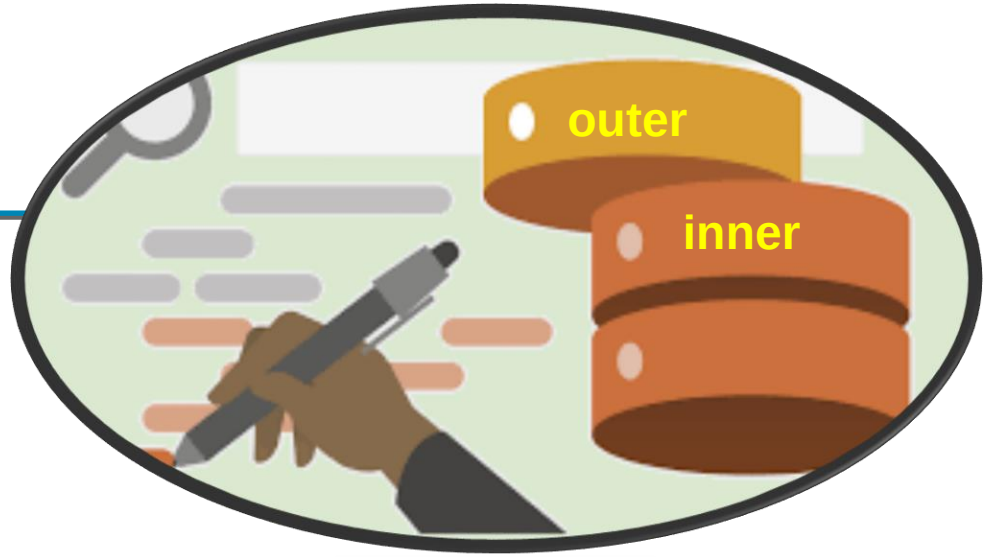
Subquery

```
Select SuccessID, SuccessName, SuccessDate, HappinessID
from MySuccess_T
where MySuccess_T.HappinessID in
(Select MyLife_T.HappinessID from MyLife_T where COVID_19 = 'Gone');
```

Result table

SuccessID	SuccessName	SuccessDate	HappinessID
1967	Got HD Grade in PF	8/10/2019	1755
2055	Got HD Grade in DF	null	1755

Objectives:



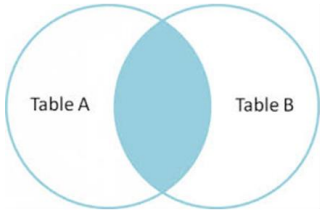
1. Non-correlated/Simple Sub Query

1.1. Using All Operator

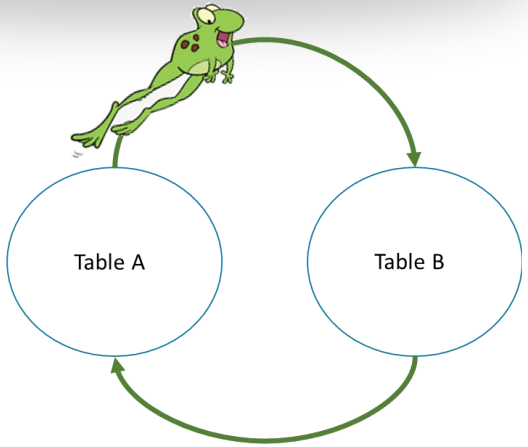
1.2. Using IN Operator

1.3. Using Any Operator

2. Join vs. Subquery



vs.



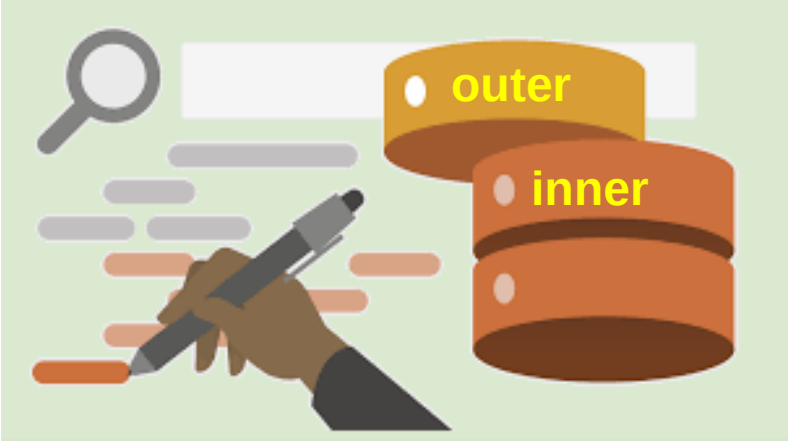
3. Examples to Write Subqueries

Processing Multiple Tables Using Subqueries

- **Subquery:** placing an **inner** query (SELECT statement) inside an **outer** query

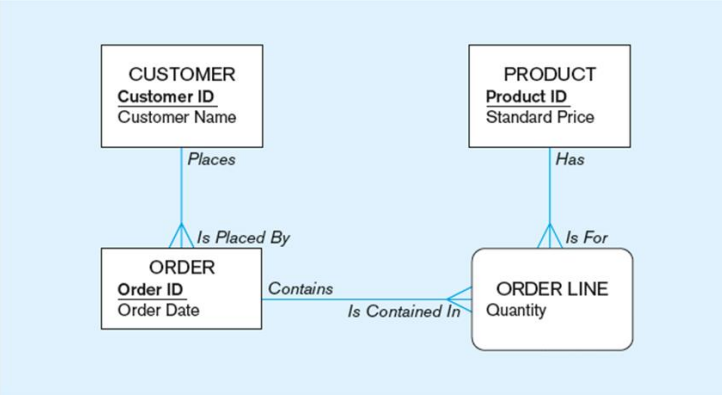
  Options:

  - In a condition of the WHERE clause
  - As a "**table**" of the FROM clause
  - Within the HAVING clause



- Subqueries (Nested queries) can be:
  - **Non-correlated (Simple or Type 1):** executed once for the entire outer query
  - **Correlated:** executed once for each row returned by the outer query

1. Simple Subquery



Question 1: What is the price of the most expensive product?

```
Select  max (productstandardprice)
from product_t;
```

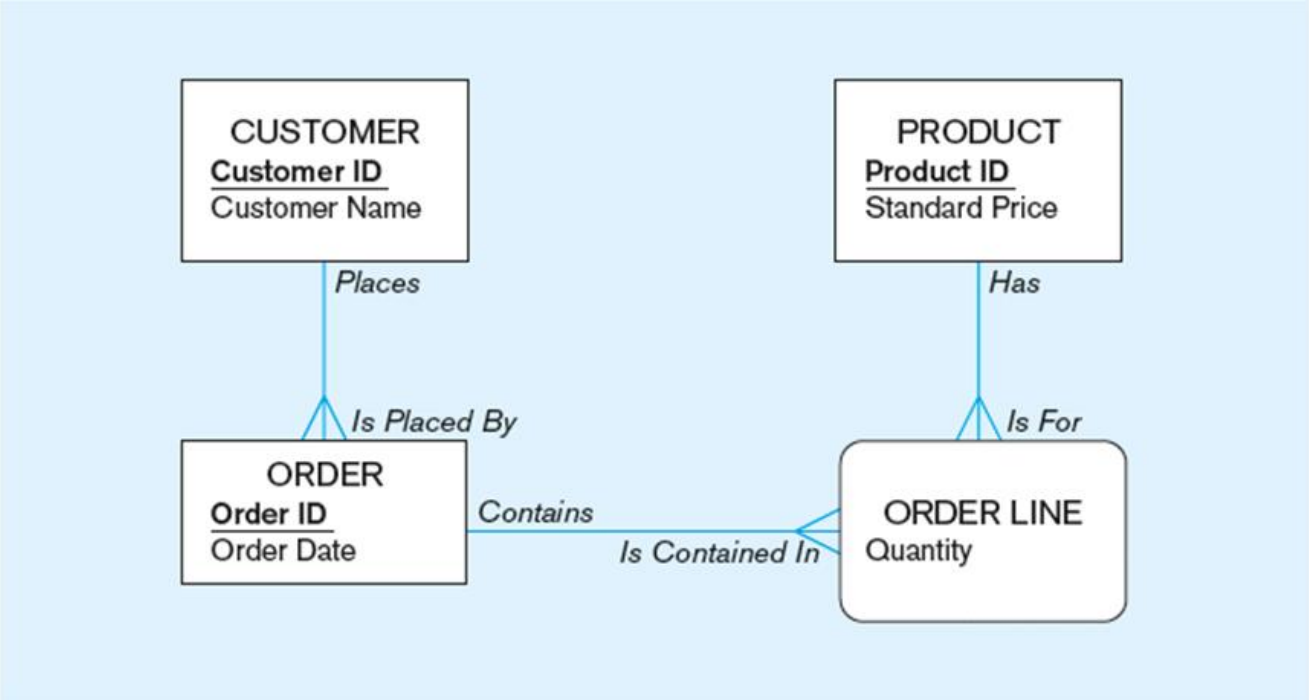
```
 max
---------
 1650.00

(1 row)
```

Note: returns a **single** value

We can get the result using a select statement with the **max aggregate function**.

What if we want to get the **product description** of most expensive product?

Question 1: What is the price of the most expensive product?



1. Simple Subquery

Question 2: What is the price and description of the most expensive product?

```
Select productdescription, productstandardprice
from product_t
where productstandardprice=1650;
```

```
 productdescription  | productstandardprice
----------------------------+----------------------
 Entertainment Center |            1650.00
(1 row)
```

Select max (productstandardprice)
from product_t;
This query can be used here as a **sub query** for the outer query)

1. Simple Subquery

Question 2: What is the price and description of the most expensive product?

```
Select productdescription, productstandardprice
from product_t
where productstandardprice= ( Select max (productstandardprice) from product_t );

 productdescription  | productstandardprice
----------------------------+----------------------
 Entertainment Center |            1650.00
(1 row)
```

1. Simple Subquery

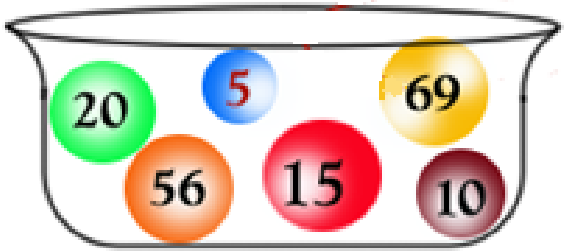
- Query inside another query
- Used in **WHERE, FROM** and **HAVING** Clauses
- **Executes once.**

```
Select productdescription, productstandardprice
from product_t
where productstandardprice= ( Select max (productstandardprice) from product_t );
```

- Most commonly used to find the **maximum** or **minimum.**

All Operator

69 >= ALL



Question: My manager needs the information about the sample with the largest number of quantities.

Sample_T

ID	Name	Quantity
22	S1	**20**
23	S2	**5**
24	S3	**69**
25	S1	**10**
26	S5	**15**
27	S6	**56**

All Operator

```
Select *
from Sample_T
where Quantity = (Select max (Quantity)
                  from Sample_T);
```

69

ID	Name	Quantity
24	S3	**69**

```
Select Quantity
from Sample_T;
```


```
Select *
from Sample_T
where Quantity > ALL (
```


```
);
```

ID	Name	Quantity
null	*null*	***null***

```
Select *
from Sample_T
where Quantity >= ALL (
```


```
);
```

ID	Name	Quantity
24	S3	**69**

```
Select *
from Sample_T
where Quantity >= ALL
```

20, 5, 69, 56, 15, 10

```
( Select Quantity
  from Sample_T);
```

ID	Name	Quantity
24	S3	**69**

1.1. Simple Subquery: Use All Operator

- The following two statements have the same results:

X= max (productstandardprice)

X>= ALL (productstandardprice)

Example:

Set A ={1, 4, 5, 8, 12, 3, 0}

X= Max (A) = 12 that means X >= all members of Set A

1.1. Simple Subquery: Use All Operator

Question 3: What is the price and description of the most expensive product?

Change the following query using ALL operator

```
Select productdescription, productstandardprice
from product_t
where productstandardprice =(Select  max (productstandardprice) from product_t);
```

```
 productdescription  | productstandardprice
----------------------------+----------------------
 Entertainment Center |            1650.00
(1 row)
```

productstandardprice = max (productstandardprice)

productstandardprice >= ALL (productstandardprice)

1.1. Simple Subquery: Use All Operator

Equivalent way to find the maximum (or minimum): "ALL"

- **Use ALL to find what is the price and description of the most expensive product?**

Select productstandardprice from product_t;

```
productstandardprice
---------------------
            175.00
            200.00
            750.00
           1650.00
            325.00
            750.00
            150.00
            175.00
            225.00
            200.00
            500.00
            800.00
            150.00
            300.00
            362.00
            890.00
           1100.00
           1200.00
            256.00
              0.00
              0.00
(21 rows)
```

Note: Similar to the earlier preliminary query, just no "max" this time, so this query returns **multiple** values.

1.1. Simple Subquery: Use All Operator

Equivalent way to find the maximum (or minimum): "ALL"

- **Use ALL to find what is the price and description of the most expensive product?**

```
Select productdescription, productstandardprice
from product_t
where productstandardprice >= ALL ( productstandardprice )
```

(The **sub query** from the previous slide goes here);

```
Select productstandardprice
from product_t;
```

```
productstandardprice
---------------------
175.00
200.00
750.00
1650.00
325.00
750.00
150.00
175.00
225.00
200.00
500.00
800.00
150.00
300.00
362.00
890.00
1100.00
1200.00
256.00
0.00
0.00
(21 rows)
```

1.1. Use All Operator

```
Select productdescription, productstandardprice
from product_t
where productstandardprice= ( Select max (productstandardprice) from product_t );
```

Equivalent way to find the maximum (or minimum): "**ALL**"

- Use ALL to find what is the price and description of the most expensive product?

```
Select productdescription, productstandardprice
from product_t
where productstandardprice >=   All( Select productstandardprice from product_t );
```

```
productdescription        | productstandardprice
---------------------------+----------------------
 Entertainment Center |            1650.00
(1 row)
```

Go to Ed

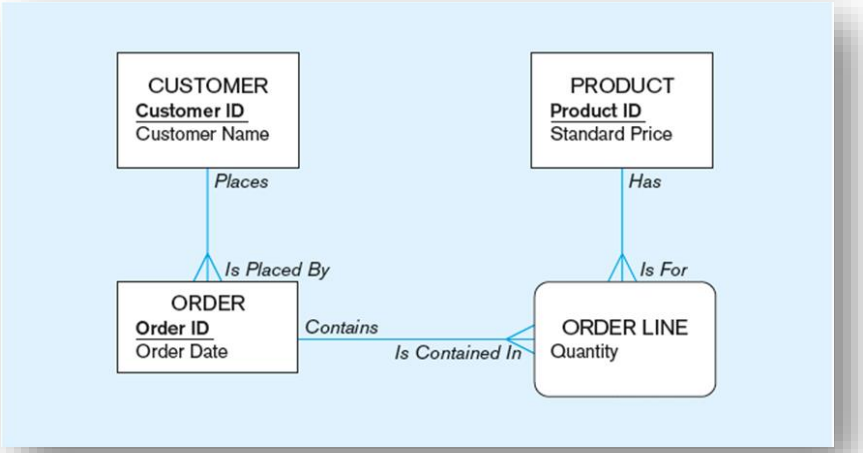
IN Operator

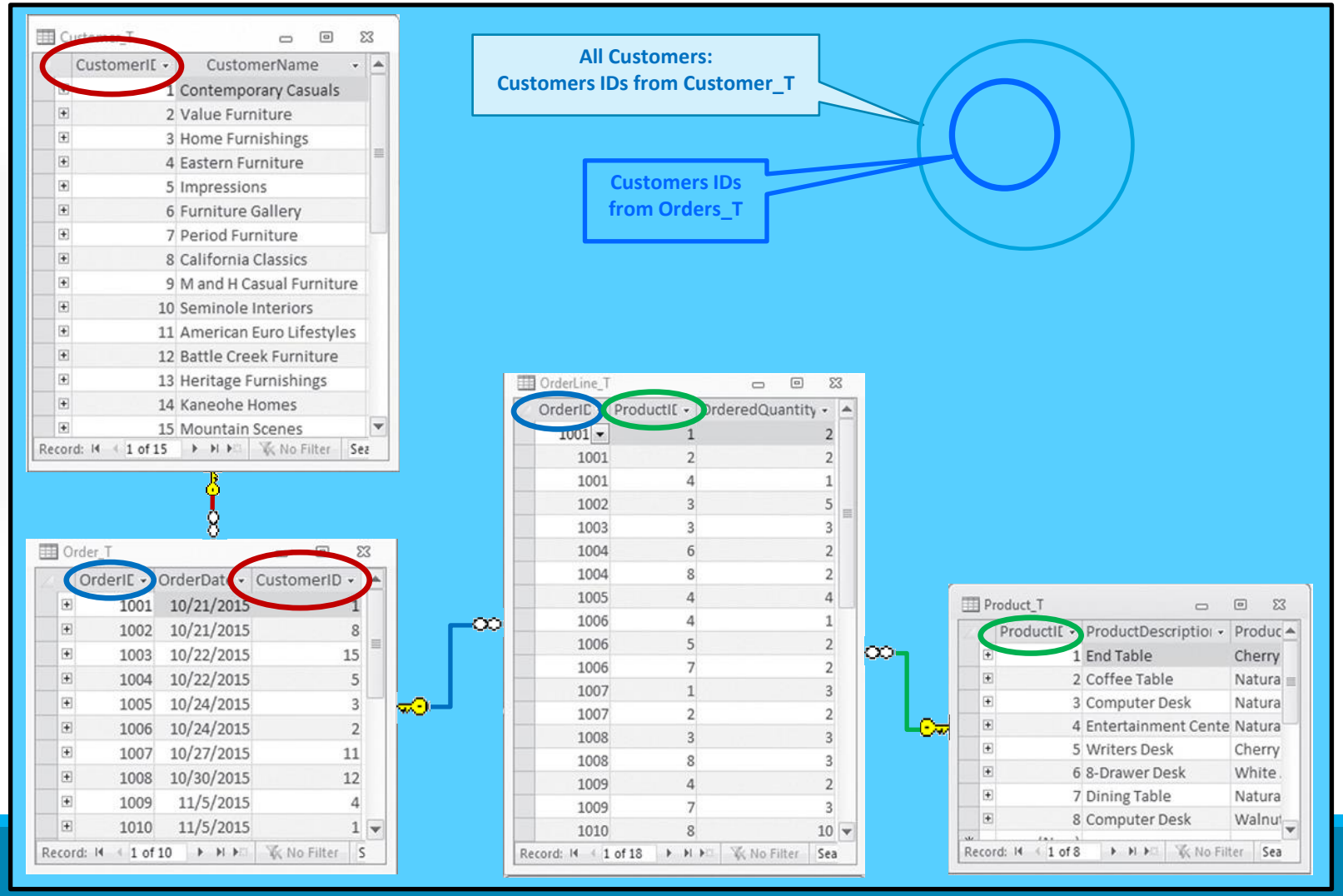
Apple is IN this set {‘Orange’, ‘Apple’, ‘Cucumber’}

Number 89 is not IN this bowl

1.2. Simple Subquery: Use IN Operator

Question 4: What are the names of customers who have placed orders?





How to write a non-correlated subquery → it is better to start with the sub-query

Question 4: What are the names of customers who have placed orders? Use IN Operator

```
Select distinct CustomerID from order_t order by CustomerID;
```

customerid
1
3
4
6
8
9
12
13
14
15
16

The IDs of customers who have placed at least one order = {1, 3, 4, 6, 8, 9, 12, 13, 14, 15, 16}

Now we need their names:

```
Select CustomerName
from    Customer_t
where CustomerID IN
                    (  Select distinct (CustomerID) from Order_t    );
```

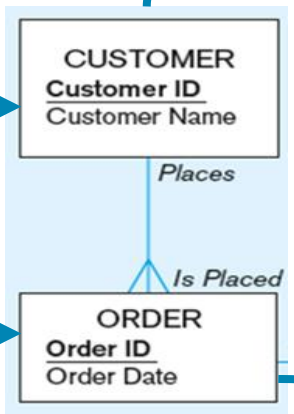
customername
Contemporary Casuals
Home Furnishings
Eastern Furniture
Furniture Gallery
Dunkins Furniture
A Carpet
Flanigan Furniture
Ikards
Wild Bills
Janet's Collection
ABC Furniture Co.

How SQL engine executes this query?
Let's check the processing of a non-correlated subquery.

Question 4: What are the names of customers who have placed orders? Use IN Operator

```
SELECT CustomerName
        FROM Customer_T
            WHERE CustomerID IN
```

```
(SELECT DISTINCT CustomerID
FROM Order_T);
```



CUSTOMERID
1
8
15
5
3
2
11
12
4

9 rows selected.

CUSTOMERNAME
Contemporary Casuals
Value Furniture
Home Furnishings
Eastern Furniture
Impressions
California Classics
American Euro Lifestyles
Battle Creek Furniture
Mountain Scenes

9 rows selected.

Go to Ed

A non-correlated subquery processes completely before the outer query begins.

Processing a non-correlated subquery: Presentation 2

```
Select CustomerName
from Customer_t
where CustomerID IN
(Select distinct(CustomerID) from Order_t);
```

① First the sub-query will be executed, and from the Ortder_T the distinct list of customer IDs will be selected.

② where CustomerID IN

③ Select CustomerName

④ from Customer_t

⑤ Select CustomerName

customerid	customername	customerstreet	customercity	customerstate	customerpostalcode
1	Contemporary Casuals	1355 S Hines Blvd	Gainesville	FL	32601-2871
2	Value Furnitures	15145 S.W. 17th St.	Plano	TX	75094-7743
3	Home Furnishings	1900 Allard Ave	Albany	NY	12209-1125
4	Eastern Furniture	1925 Beltline Rd.	Carteret	NJ	07008-3188
5	Impressions	5585 Westcott Ct.	Sacramento	CA	94206-4056
6	Furniture Gallery	325 Flatiron Dr.	Boulder	CO	80514-4432
7	New Furniture	Palace Ave	Farmington	NM	
8	Dunkins Furniture	7700 Main St	Syracuse	NY	31590
9	A Carpet	434 Abe Dr	Rome	NY	13440
12	Flanigan Furniture	Snow Flake Rd	Ft Walton Beach	FL	32548
13	Ikards	1011 S. Main St	Las Cruces	NM	88001
14	Wild Bills	Four Horse Rd	Oak Brook	Il	60522
15	Janet's Collection	Janet Lane	Virginia Beach	VA	10012
16	ABC Furniture Co.	152 Geramino Drive	Rome	NY	13440

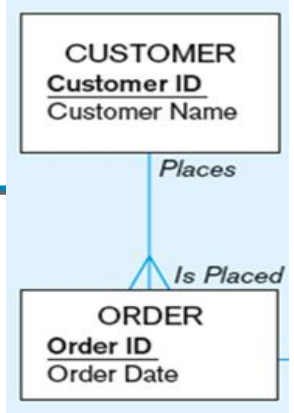
customerid
1
3
4
6
8
9
12
13
14
15
16

The result table of the sub-query

customername
Contemporary Casuals
Home Furnishings
Eastern Furniture
Furniture Gallery
Dunkins Furniture
A Carpet
Flanigan Furniture
Ikards
Wild Bills
Janet's Collection
ABC Furniture Co.

The result table of the outer query

Question 4: What are the names of customers who have placed orders? Use IN Operator




```
SELECT CustomerName
  FROM Customer_T
    WHERE CustomerID IN
      (SELECT DISTINCT CustomerID
      FROM Order_T);
```

The **IN** operator will test to see if the CUSTOMER_ID value of **a row** is included in the list returned from the subquery

Subquery is embedded in parentheses. In this case it returns a list that will be used in the WHERE clause of the outer query

Sup-query Results:

CUSTOMERID
1
8
15
5
3
2
11
12
4

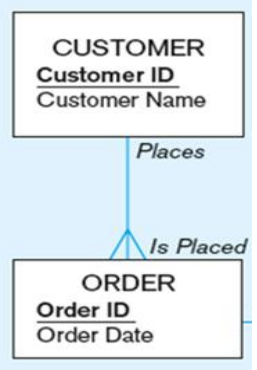
9 rows selected.

Outer Query Results:

CUSTOMER_NAME
Contemporary Casuals
Value Furniture
Home Furnishings
Eastern Furniture
Impressions
California Classics
American Euro Lifestyles
Battle Creek Furniture
Mountain Scenes

9 rows selected.

1.3. Simple Subquery: Use Any Operator



Question 4: What are the names of customers who have placed orders?

```
Select CustomerName
from Customer_t
where CustomerID = ANY
    (Select distinct(CustomerID)
     from Order_t);
```

The **ANY** operator will test to see if the CUSTOMER_ID value of **a row** is included in the list returned from the subquery

Sup-query Results:

CUSTOMERID
1
8
15
5
3
2
11
12
4

9 rows selected.

Outer Query Results:

CUSTOMER_NAME
Contemporary Casuals
Value Furniture
Home Furnishings
Eastern Furniture
Impressions
California Classics
American Euro Lifestyles
Battle Creek Furniture
Mountain Scenes

9 rows selected.

1.4. Rules for using ALL, Any and IN

When a subquery **returns more than one value**	**ANY**, **IN** or **ALL** will be used to link the subquery and main query.
When **ALL** is used	**every** value returned by the subquery must satisfy the comparison operation.
When **ANY** or **IN** is used	**one or more** values returned by the subquery must satisfy the comparison operation.

IN → **is equivalent to** → **= ANY**

Not IN → **is equivalent to** → **<> ALL**

NOTEs:

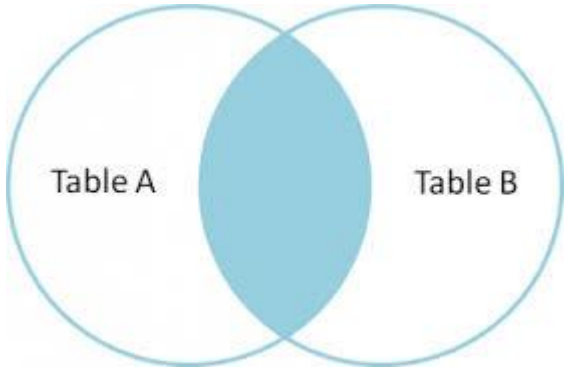
1- You can use comparison operators (<, >, <>) with **ANY** and **ALL** But not with **IN**.

2- IN can take a subquery or a list of **values** as parameters, but **ANY** and **ALL** operators expects an **array** or a subquery.

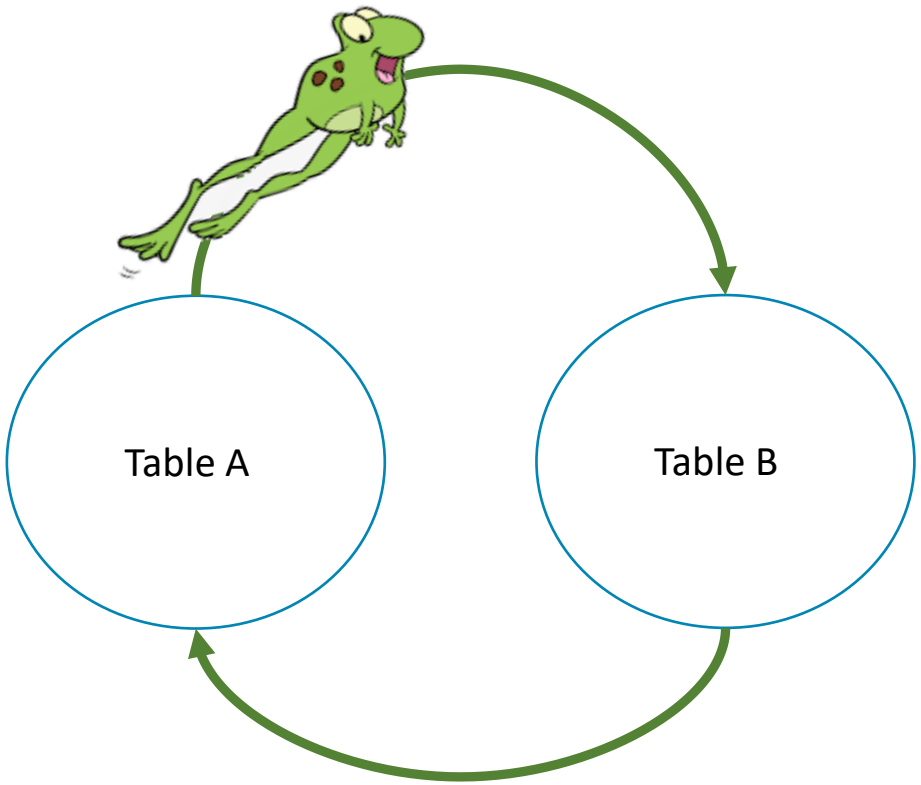
I prefer to avoid “ALL” and “ANY”.)

2. Join vs. Subquery

Some queries could be accomplished by either a join or a subquery

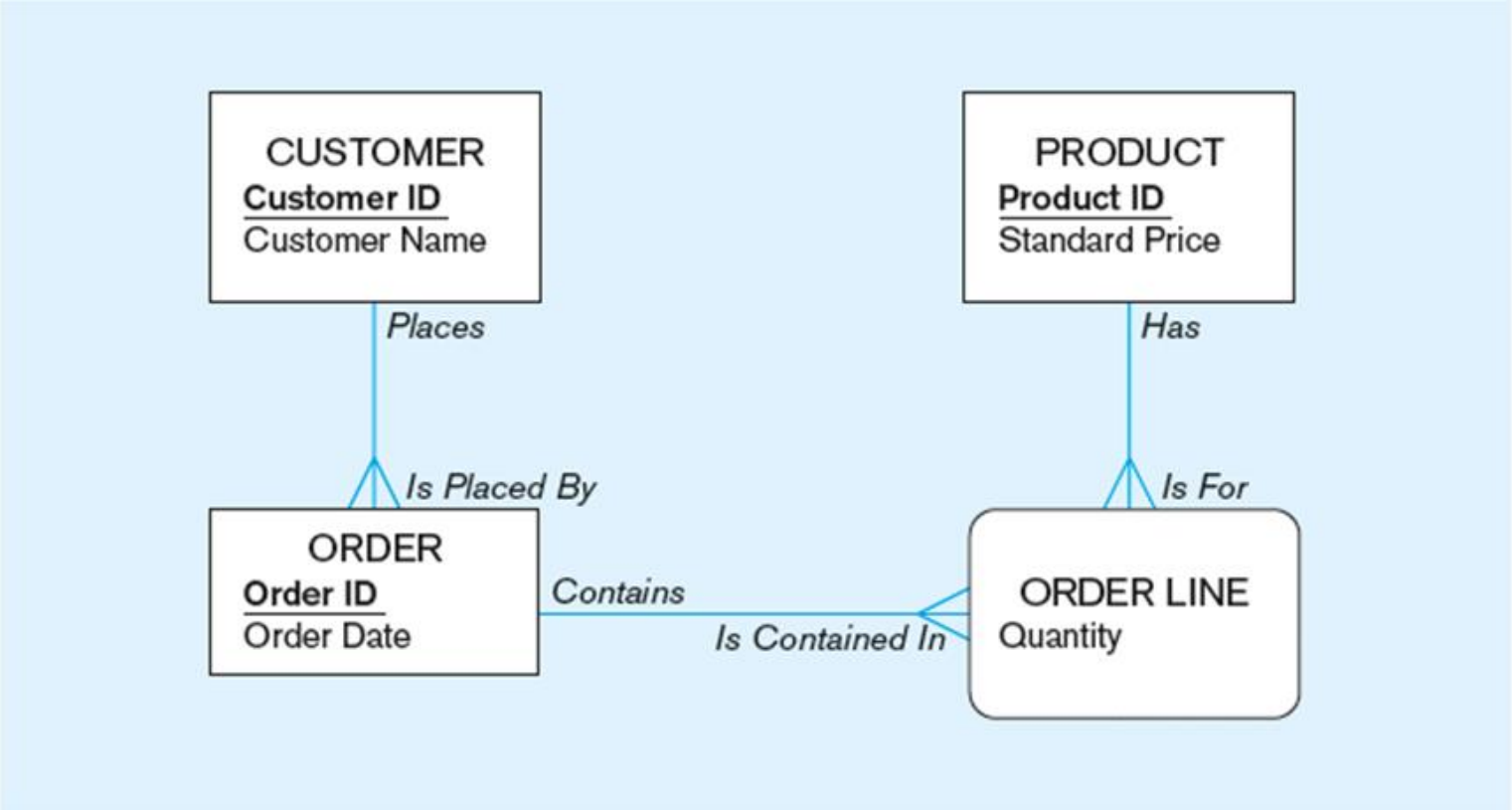


vs.

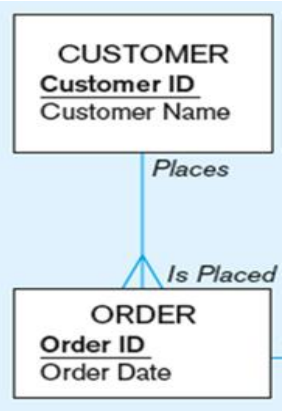


2. Join vs. Subquery

Question 5: What are the name and address of the customer who placed order number 20?



2. Join vs. Subquery



Question 5: What are the name and address of the customer who placed order number 20?

Join version

```
SELECT CustomerName, CustomerAddress, CustomerCity,
   CustomerState, CustomerPostalCode
FROM Customer_T, Order_T
WHERE Customer_T.CustomerID = Order_T. CustomerID
   AND OrderID = 20 ;
```

Subquery version

```
SELECT CustomerName, CustomerAddress, CustomerCity,
CustomerState, CustomerPostalCode
   FROM Customer_T
      WHERE Customer_T.CustomerID =
         (SELECT Order_T.CustomerID
            FROM Order_T
               WHERE OrderID = 20 );
```

Graphical depiction of a query using join

- Question 5:What are the name and address of the customer who placed order number 20?

```
SELECT CustomerName, CustomerStreet , CustomerCity,
   CustomerState, CustomerPostalCode
FROM Customer_T, Order_T
WHERE Customer_T.CustomerID = Order_T. CustomerID
   AND OrderID =  20 ;
```

WHERE Order_T.CustomerID = Customer_T.CustomerID

WHERE OrderID = **20**

CUSTOMER_T

CustomerID	Customer Name	Customer Street	Customer City	Customer State	Customer PostalCode
3	...	...	...	...	...
4	...	...	...	...	...
5	...	...	...	...	...
	...	...	...	...	...

ORDER_T

CustomerID	OrderID	....
3	10	....
4	20	....
6	20	....
....	....	....

Graphical depiction of a query using simple subquery.

➢ Question 5:What are the name and address of the customer who placed order number 20?

```
SELECT CustomerName, CustomerStreet , CustomerCity,
CustomerState, CustomerPostalCode
  FROM Customer_T
    WHERE Customer_T.CustomerID =
      (SELECT Order_T.CustomerID
        FROM Order_T
          WHERE OrderID = 20 );
```

1 2 3 4 5

Outer-Query results the required information of the customer who has placed OrderID=20

Sub-Query results the CustomerID from **Order_T Where OrderID=20 (i.e. 4 in this example)**

```
SELECT  Order_T.CustomerID
  FROM Order_T
    WHERE OrderID = 20;
```

```
SELECT  Customer_T.customerid, customername, customerstreet, customercity, customerstate, customerpostalcode
FROM    Customer_T
WHERE  Customer_T.customerid = (SELECT  Order_T.CustomerID  FROM Order_T  WHERE OrderID = 20);
```

ORDER_T

CustomerID	OrderID	....
3	10	....
4	20	....
6	22	....
....	....	....

CUSTOMER_T

CustomerID	Customer Name	Customer Street	Customer City	Customer State	Customer PostalCode
3	...	...	...	...	...
4	Eastern Furniture	1925 Beltline Rd.	Carteret	NJ	07008-3188
5	...	...	...	...	...
	...	...	...	...	...

Go to Ed

Video

How to use the SQL In Statement with Subquery

Link:

When we have to use Join or Subquery?

When just Join? When just sub-query?

Usually sub-query is more efficient than join, however, sometimes we have to use join instead.

Just use Join:

- When you need data from both tables

Just use sub-query:

- When you need to find a data with the largest/smallest value for an attribute in the database

Note: When you have composite PK, you need to check the equality of the composite PK and FK in your query using join or subquery:

```
select patadmdate
from patmchart
where (patid, patcid) in  (select patid, patcid  from prescribeddrug  where …. );

Select patadmdate
from patmchart p inner join  prescribeddrug pr  on  p.patid= pr.patid and p.patcid=pr.patcid
Where …. ;

Select patadmdate
from patmchart p inner join  prescribeddrug pr  on  (p.patid, p.patcid) = (pr.patid, pr.patcid)
Where …. ;
```

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 success** after **COVID-19 gone**.

MyLife_T

HappinessID	HappinessName	HppinessStartDate	HppinessEndDate	COVID-19
1755	Pass DF	09/03/2020	Soon	Gone
1899	Graduated	09/03/2019	Soon	Came
…	…	…	…	…

MySuccess_T

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

Join

```
Select SuccessID, SuccessName, SuccessDate, MyLife_T.HappinessID
from MyLife_T Inner Join MySuccess_T on MyLife_T.HappinessID = MySuccess_T. HappinessID
where COVID_19 = 'Gone';
```

Subquery

```
Select * from MySuccess_T
where MySuccess_T.HappinessID in
(Select MyLife_T.HappinessID from MyLife_T where COVID_19 = 'Gone');
```

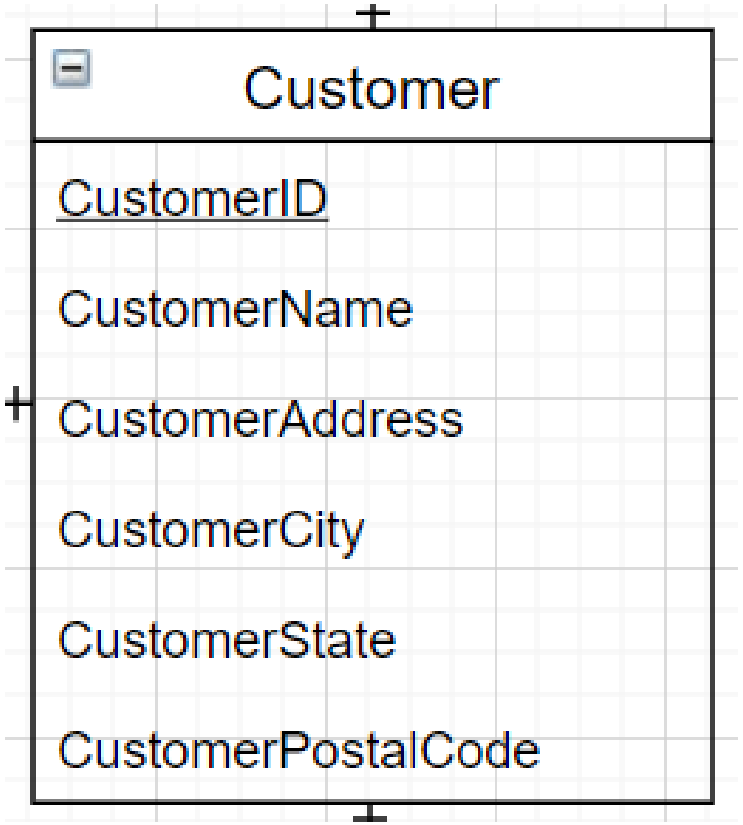
Result table

SuccessID	SuccessName	SuccessDate	HappinessID
1967	Got HD Grade in PF	8/10/2019	1755
2055	Got HD Grade in DF	Soon	1755

Examples

Example 1

- Display the name of customer 16 and the names of all the customers that are in the same postal code as customer 16.



Example 1

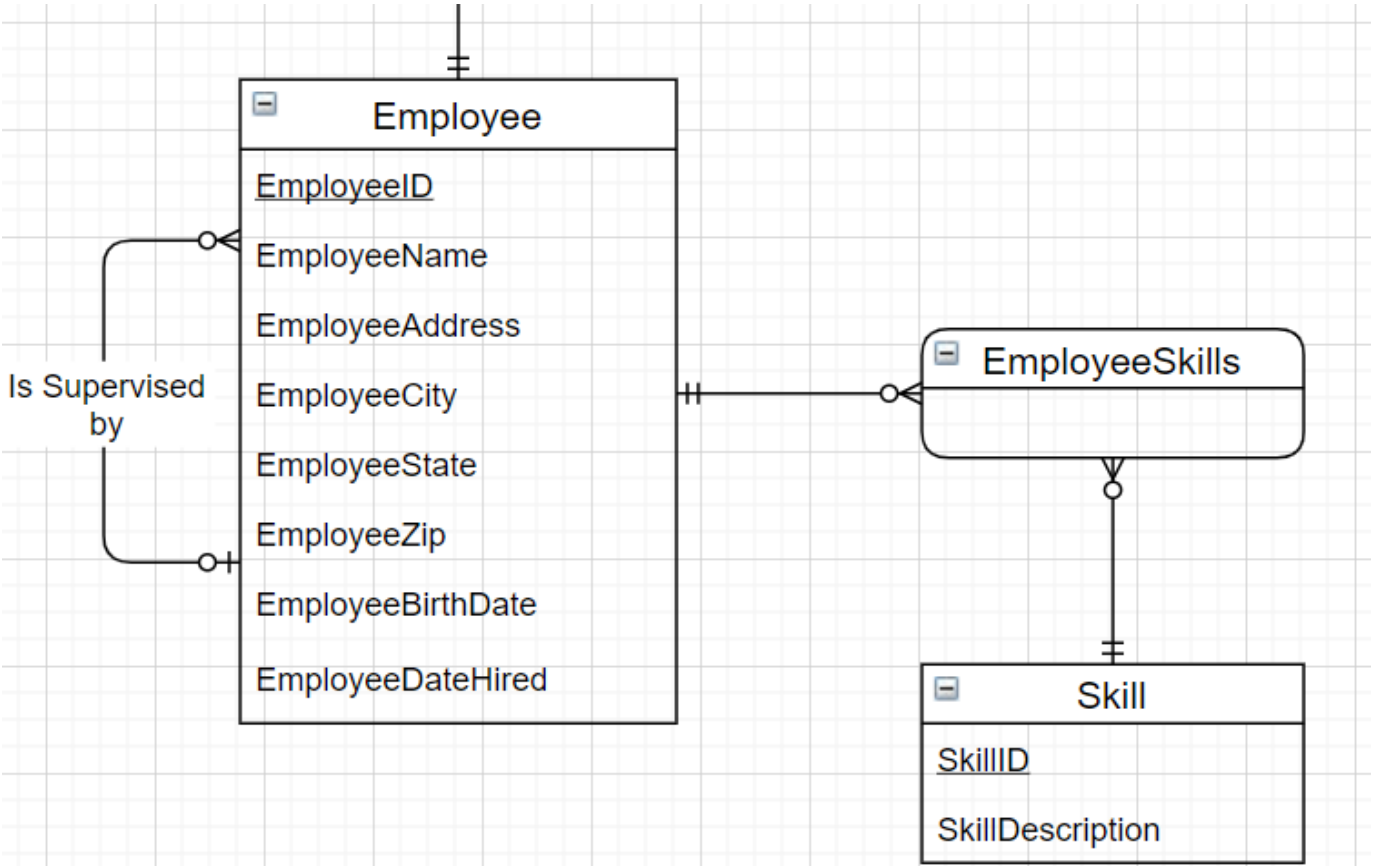
- **Display the name of customer 16 and the names of all the customers that are in the same postal code as customer 16.**

```
select customerid, customername,customerpostalcode
from customer_t
where customerpostalcode=
                                        (select customerpostalcode
                                         from customer_t
                                         where customerid=16);


 customerid |   customername    | customerpostalcode
--------------+---------------- -----+--------------------
         9        | A Carpet              | 13440
        16        | ABC Furniture Co. | 13440
(2 rows)
```

Example 2

- List names of all employees who are **managing** people with **skill ID BS12**, list each manger's name only once, even if that manger manages several people with this skill.

  - 1- Employee with skill ID BS12
  - 2- IDs of their managers
  - 3- The names of their managers

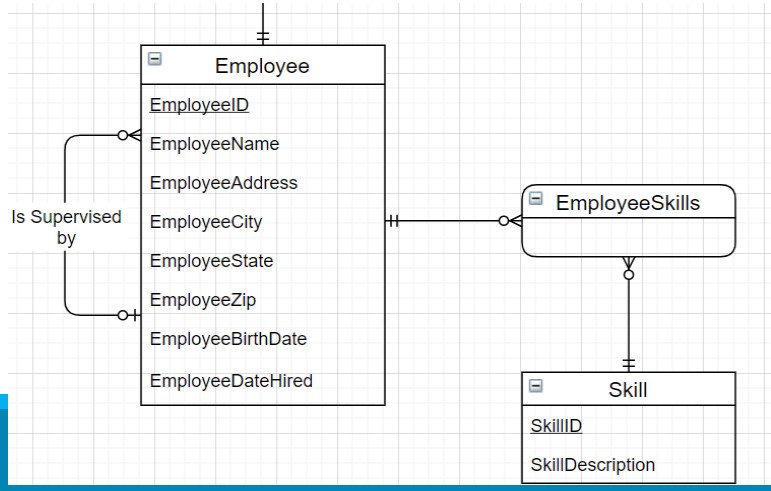


Example 2 - Solution 1: using nested subqueries

- List names of all employees who are managing people with skill ID BS12, list each manger's name only once, even if that manger manages several people with this skill.

```
select m.employeename
from employee_t m
where m.employeeid in
                    (select e.employeesupervisor from employee_t e
                     where e.employeeid in
                     (select employeeid from employeeskills_t where skillid='BS12'));
```

```
employeename
--------------
 Robert Lewis
 Phil Morris
(2 rows)
```

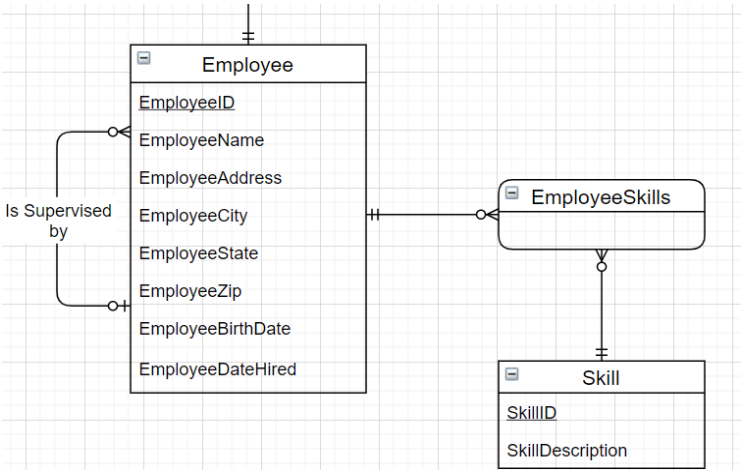


Example 2-Solution 2: using the combination of self-join and subquery

- List names of all employees who are managing people with skill ID BS12, list each manger's name only once, even if that manger manages several people with this skill.

```
select M.employeename
from employee_t E,employee_t M
where E.employeesupervisor=M.employeeid
        and E.employeeid in
          (select employeeid from employeeskills_t where skillid='BS12');
```

```
employeename
--------------
 Robert Lewis
 Phil Morris
(2 rows)
```

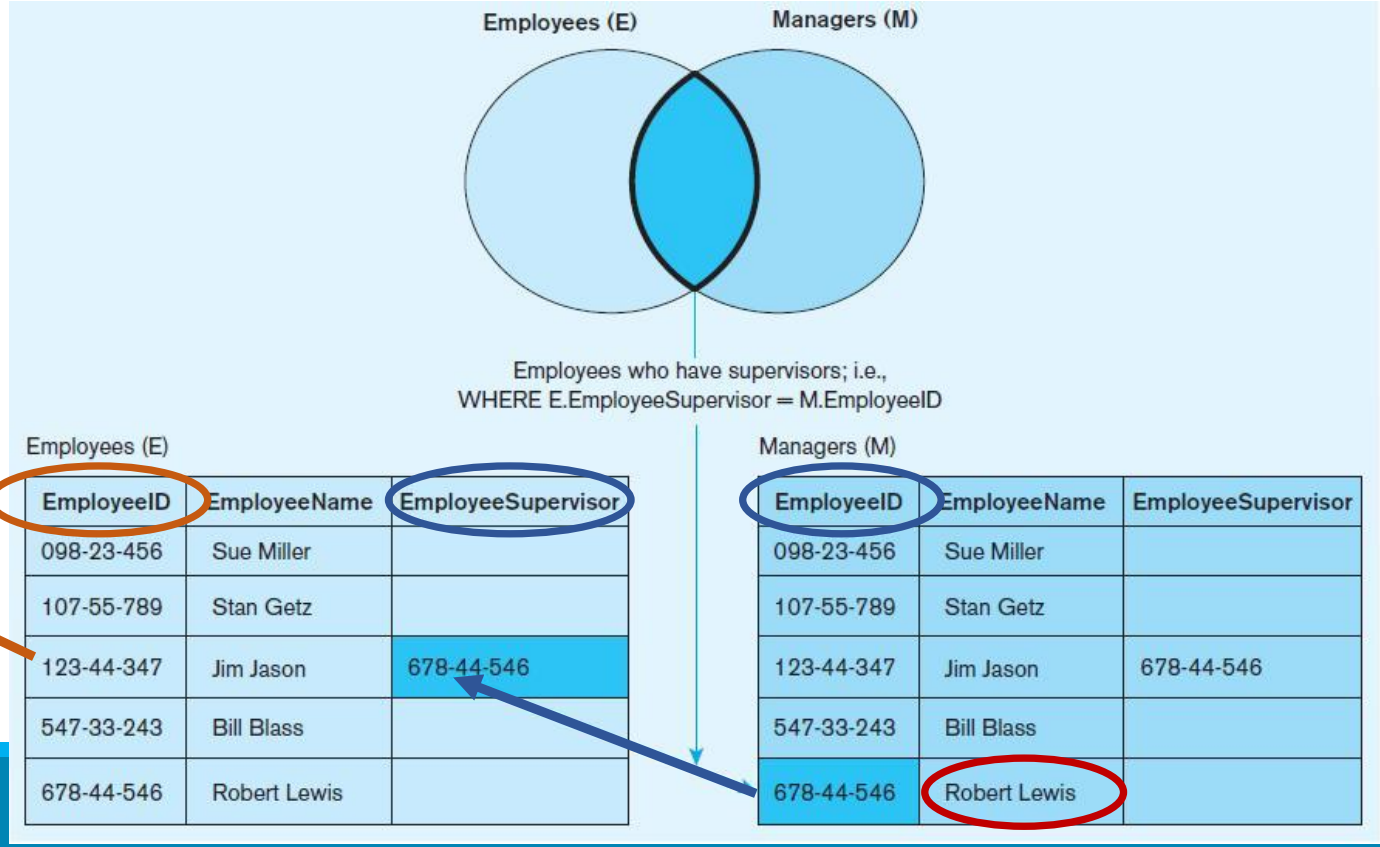


Example 2-Solution 2: using the combination of self-join and subquery

```
select M.employeename from employee_t E,employee_t M
where E.employeesupervisor=M.employeeid and
E.employeeid  in (select employeeid from employeeskills_t where skillid='BS12');
```

EmployeeSkill

EmployeeID	SkillID	Grade
123-44-347	**BS12**	Pass
107-55-789	CV12	Pass



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	

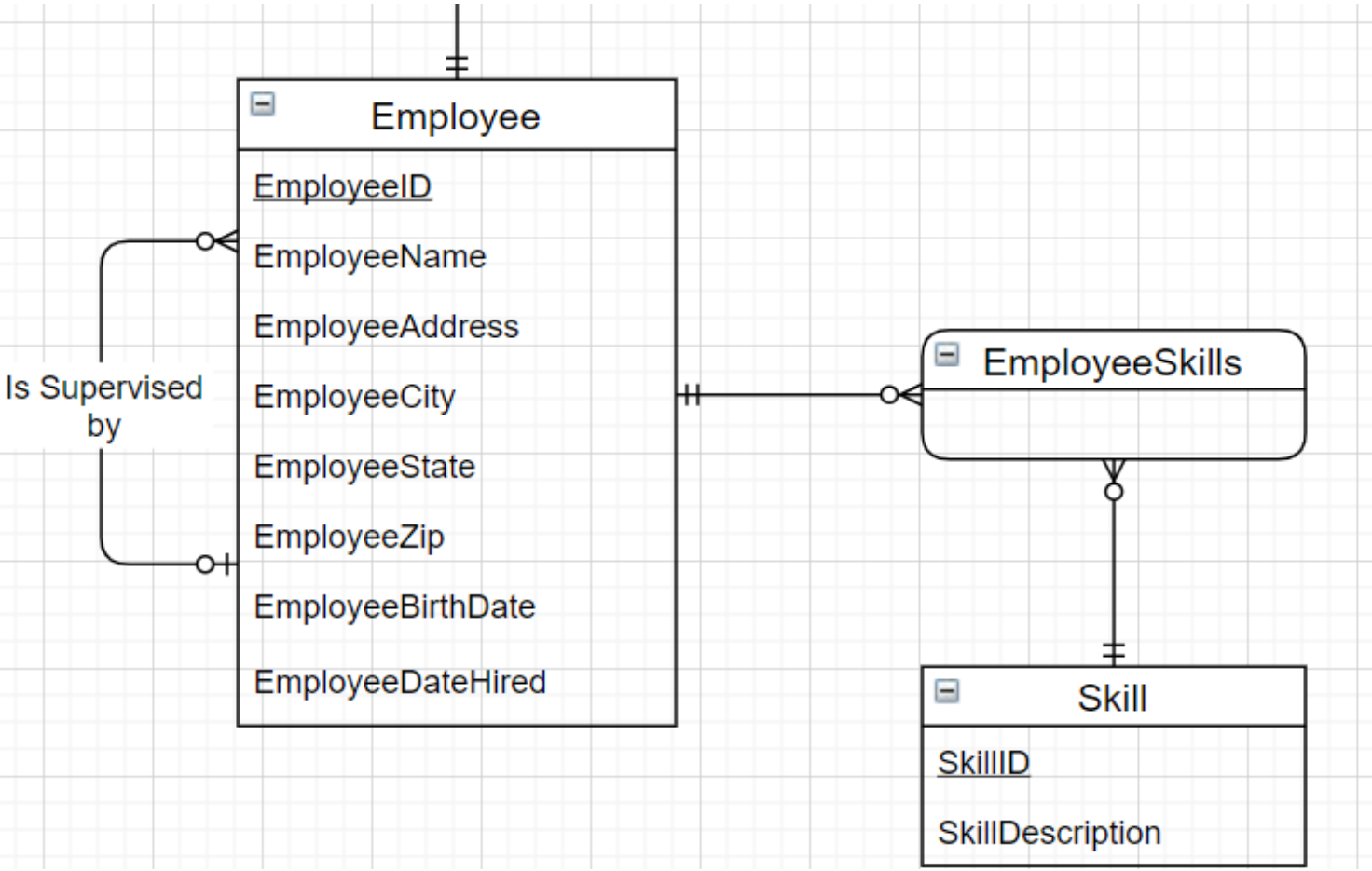
EmployeeID	EmployeeName	EmployeeSupervisor
098-23-456	Sue Miller	
107-55-789	Stan Getz	
123-44-347	Jim Jason	678-44-546
547-33-243	Bill Blass	
678-44-546	Robert Lewis	

(Figure 7-5)

Example 3

- Display the employee ID and name for those employees who do not possess the skill Router.

- 1- skill ID
- 2- People with that skill
- 3- people who has not that skill

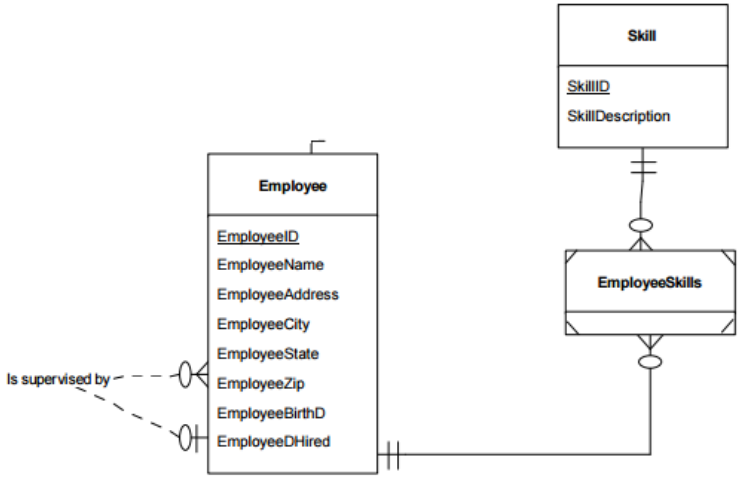


Example 3

- Display the employee ID and name for those employees who do not possess the skill Router.

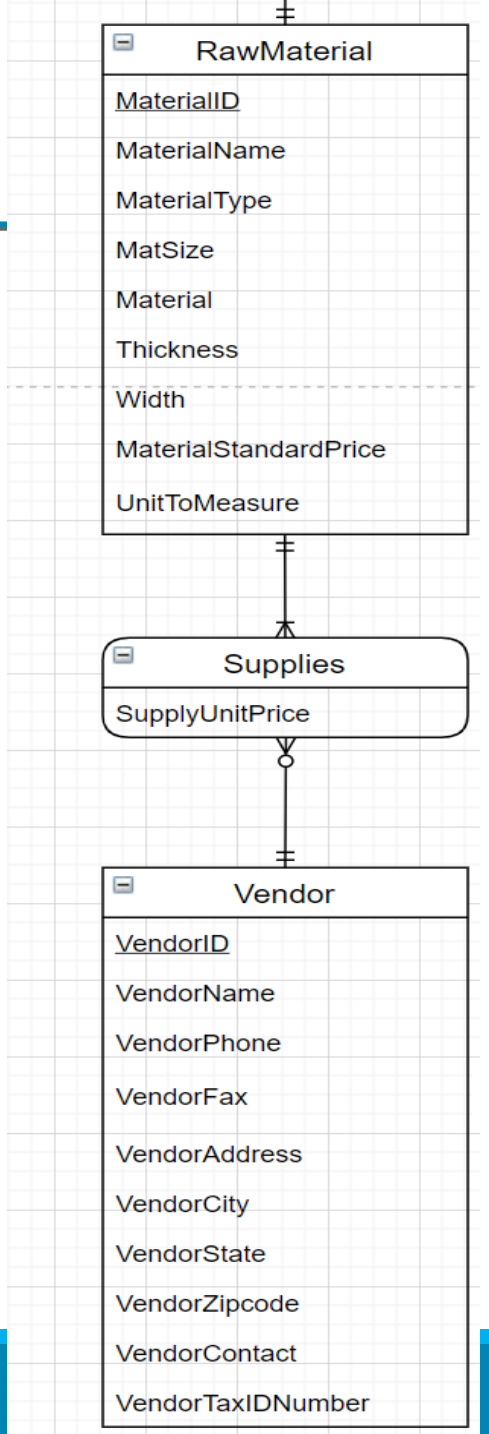
```
select employeeid, employeename from employee_t

where employeeid not in

                    (select employeeid from employeeskills_t

                     where skillid = (select skillid from skill_t where skilldescription='Router'));
```

```
employeeid |  employeename
---------------+-----------------
 123-44-345 | Phil Morris
 332445667  | Lawrence Haley
 555955585  | Mary Smith
 265955585  | Laura Ellenburg
(4 rows)
```



Example 4

- **Display How many raw materials are supplied by Pebbles Hardware?**

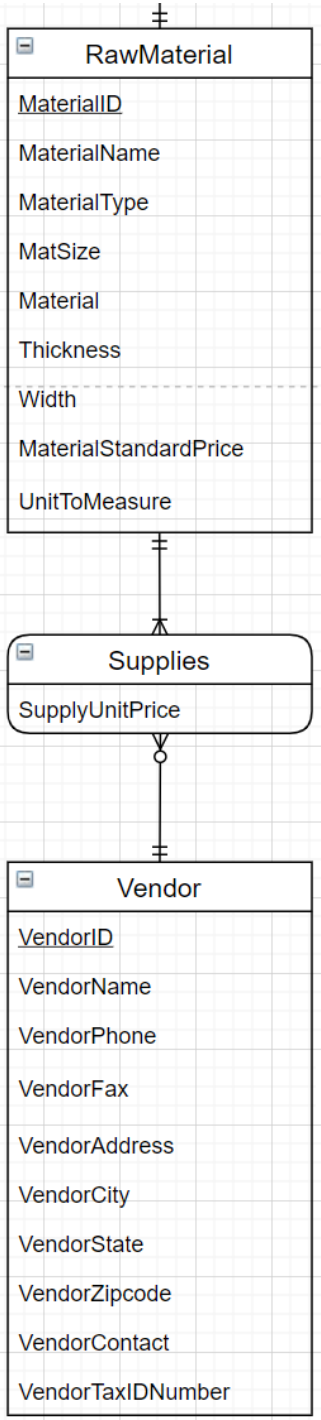


Example 4

- Display How many raw materials are supplied by Pebbles Hardware?

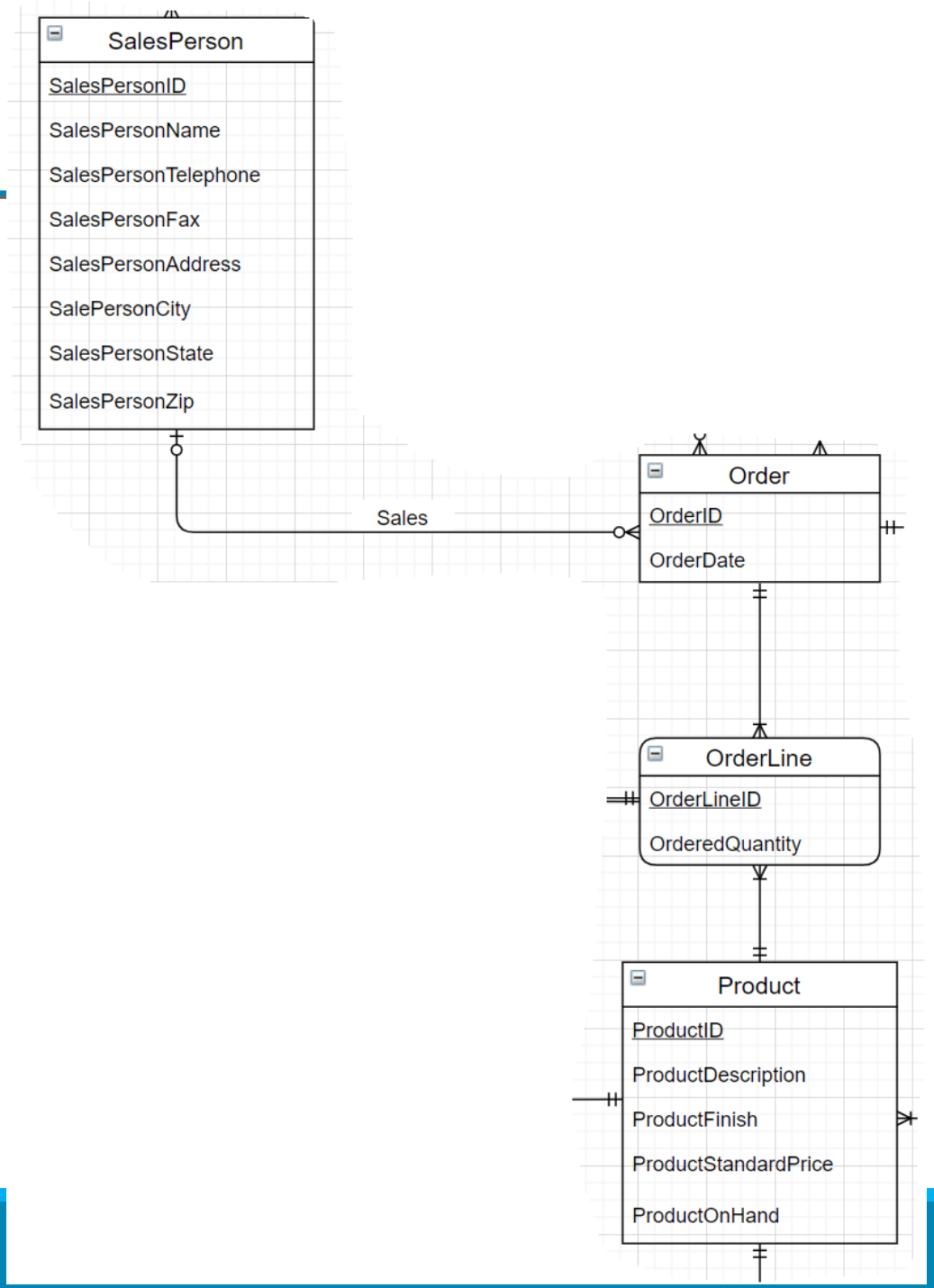
```
select count(Materialid)
from supplies_t
where vendorid=
        (select vendorid from vendor_t where vendorname='Pebbles Hardware');
```

```
count
-------
  127
(1 row)
```



Example 5

- Calculate the number of computer desks sold by each salesperson.

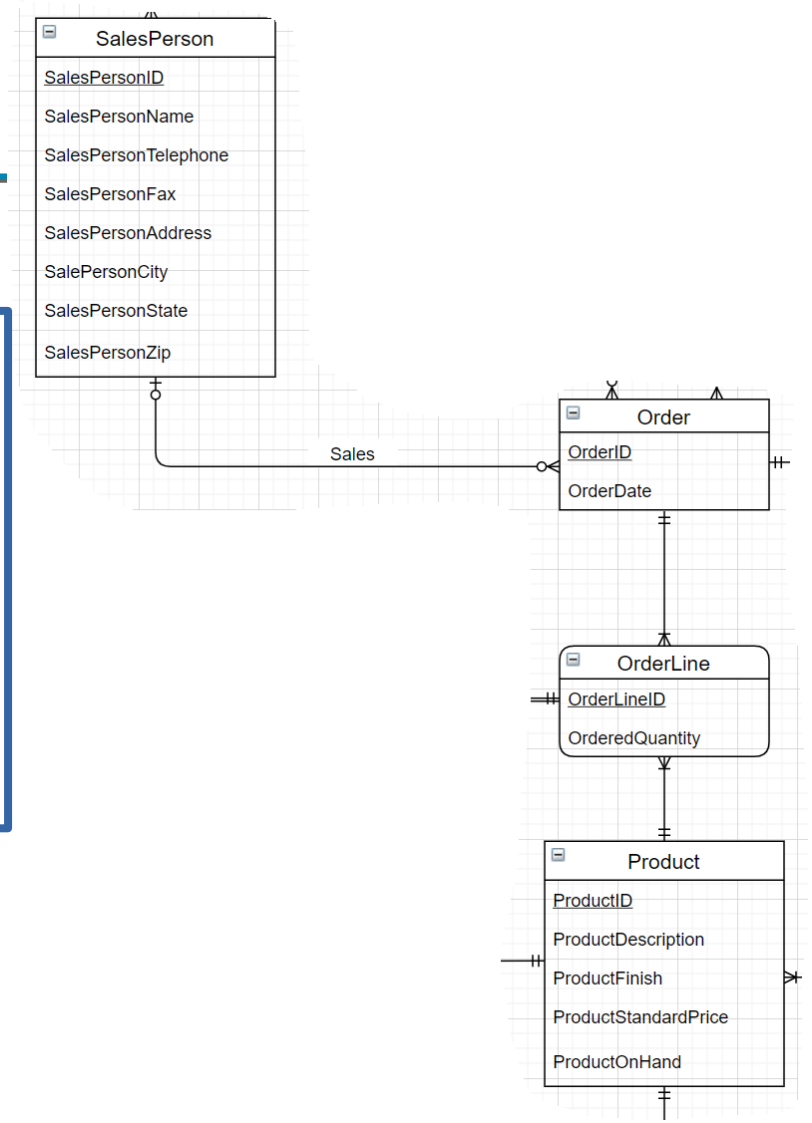


Example 5

- Calculate the number of computer desks sold by each salesperson.

```
select salespersonid, sum(orderedquantity)
from order_t O inner join orderline_t  OL
        on O.ordered = OL.ordered
        where productid=
                (select productid from product_t
                where productdescription like '%Computer Desk%')
group by salespersonid;
```

```
salespersonid | sum
---------------+-----
             3 |  12
             5 |   2
             6 |   5
             9 |   3
               |   3
(5 rows)
```



Note: if you want to remove the row where salespersonid has a *null* value, you can revise your query as follows:

```
select salespersonid, sum(orderedquantity)
from order_t O inner join orderline_t  OL
        on O.ordered = OL.ordered
where productid=
                (select productid from product_t
                where productdescription like '%Computer Desk%')
group by salespersonid
having salespersonid is not null;
```

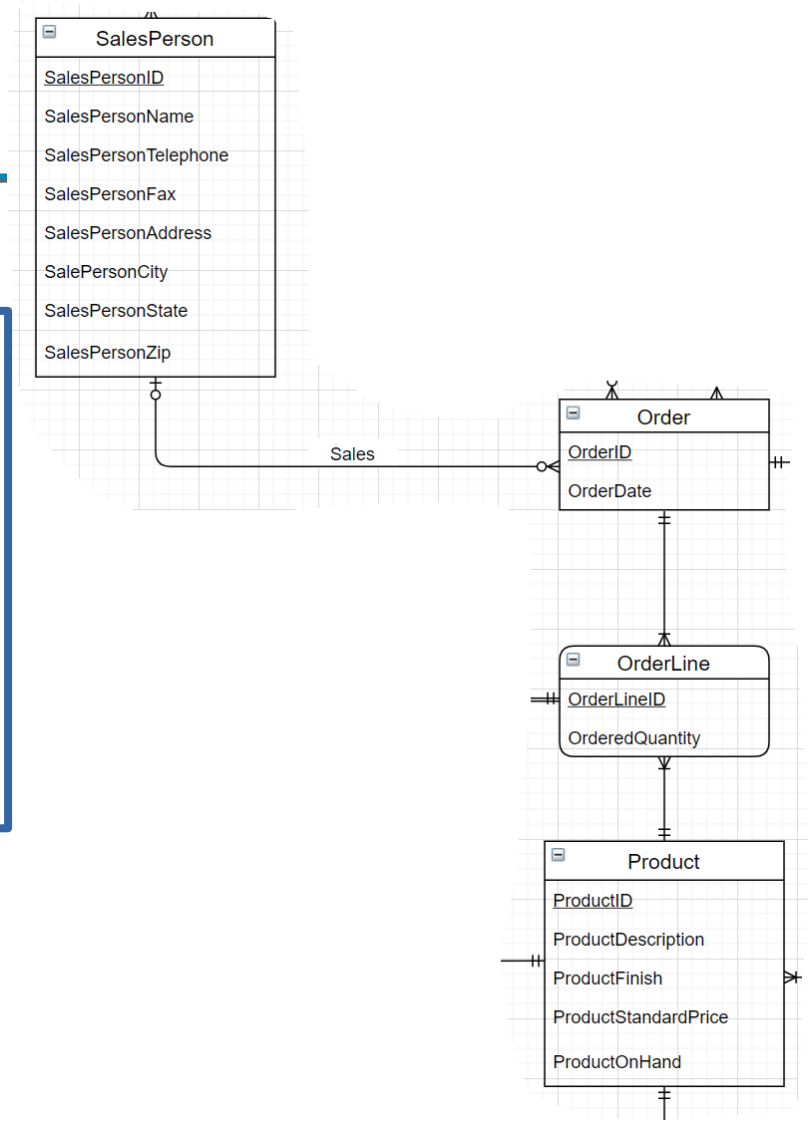
```
select salespersonid, sum(orderedquantity)
from order_t O inner join orderline_t  OL
        on O.ordered = OL.ordered
        where productid=
                (select productid from product_t
                where productdescription like '%Computer Desk%')
        and salespersonid is not null
group by salespersonid;
```

```
salespersonid | sum
---------------+-----
            3 |  12
            5 |   2
            6 |   5
            9 |   3
(4 rows)
```

Example 5 (using Natural join)

- **Calculate the number of computer desks sold by each salesperson.**

```
select salespersonid, sum(orderedquantity)
from order_t O natural join orderline_t  OL
        where productid=
                (select productid from product_t
                where productdescription like '%Computer Desk%')
group by salespersonid
having salespersonid is not null;
```




```
salespersonid | sum
---------------+-----
             3 |  12
             5 |   2
             6 |   5
             9 |   3
(4 rows)
```