

# IN2090 – Databaser og datamodellering

## 10 – Ytre joins

Leif Harald Karlsen  
leifhka@ifi.uio.no



Universitetet i Oslo

# Aggregering og NULL

---

- ◆ Aggregering med `sum`, `min`, `max` og `avg` ignorerer `NULL`-verdier

## Aggregering og NULL

---

- ◆ Aggregering med `sum`, `min`, `max` og `avg` ignorerer `NULL`-verdier
- ◆ Det betyr også at dersom det kun er `NULL`-verdier i en kolonne blir resultatet av disse `NULL`

## Aggregering og NULL

---

- ◆ Aggregering med `sum`, `min`, `max` og `avg` ignorerer `NULL`-verdier
- ◆ Det betyr også at dersom det kun er `NULL`-verdier i en kolonne blir resultatet av disse `NULL`
- ◆ `count (*)` teller med `NULL`-verdier

## Aggregering og NULL

---

- ◆ Aggregering med `sum`, `min`, `max` og `avg` ignorerer `NULL`-verdier
- ◆ Det betyr også at dersom det kun er `NULL`-verdier i en kolonne blir resultatet av disse `NULL`
- ◆ `count (*)` teller med `NULL`-verdier
- ◆ Men dersom vi oppgir en konkret kolonne, f.eks. `count (product_name)` vil den kun telle verdiene som ikke er `NULL`

# Aggregering og NULL

- ◆ Aggregering med `sum`, `min`, `max` og `avg` ignorerer `NULL`-verdier
- ◆ Det betyr også at dersom det kun er `NULL`-verdier i en kolonne blir resultatet av disse `NULL`
- ◆ `count(*)` teller med `NULL`-verdier
- ◆ Men dersom vi oppgir en konkret kolonne, f.eks. `count(product_name)` vil den kun telle verdiene som ikke er `NULL`
- ◆ For eksempel:

Name	Age
Per	2
Kari	4
Mari	NULL

```
SELECT min(Age) FROM Person;  
SELECT avg(Age) FROM Person;  
SELECT count(Age) FROM Person;  
SELECT count(*) FROM Person;
```

```
SELECT sum(Age) FROM Person  
WHERE Name = 'Mari';
```

```
SELECT count(Age) FROM Person  
WHERE Name = 'Mari';
```

# Aggregering og NULL

- ◆ Aggregering med `sum`, `min`, `max` og `avg` ignorerer `NULL`-verdier
- ◆ Det betyr også at dersom det kun er `NULL`-verdier i en kolonne blir resultatet av disse `NULL`
- ◆ `count(*)` teller med `NULL`-verdier
- ◆ Men dersom vi oppgir en konkret kolonne, f.eks. `count(product_name)` vil den kun telle verdiene som ikke er `NULL`
- ◆ For eksempel:

Name	Age
Per	2
Kari	4
Mari	NULL

```
SELECT min(Age) FROM Person;    --> 2
SELECT avg(Age) FROM Person;
SELECT count(Age) FROM Person;
SELECT count(*) FROM Person;
```

```
SELECT sum(Age) FROM Person
WHERE Name = 'Mari';
```

```
SELECT count(Age) FROM Person
WHERE Name = 'Mari';
```

# Aggregering og NULL

- ◆ Aggregering med `sum`, `min`, `max` og `avg` ignorerer `NULL`-verdier
- ◆ Det betyr også at dersom det kun er `NULL`-verdier i en kolonne blir resultatet av disse `NULL`
- ◆ `count(*)` teller med `NULL`-verdier
- ◆ Men dersom vi oppgir en konkret kolonne, f.eks. `count(product_name)` vil den kun telle verdiene som ikke er `NULL`
- ◆ For eksempel:

Name	Age
Per	2
Kari	4
Mari	NULL

```
SELECT min(Age) FROM Person;    --> 2
SELECT avg(Age) FROM Person;    --> 3
SELECT count(Age) FROM Person;
SELECT count(*) FROM Person;
```

```
SELECT sum(Age) FROM Person
WHERE Name = 'Mari';
```

```
SELECT count(Age) FROM Person
WHERE Name = 'Mari';
```



# Aggregering og NULL

- ◆ Aggregering med `sum`, `min`, `max` og `avg` ignorerer `NULL`-verdier
- ◆ Det betyr også at dersom det kun er `NULL`-verdier i en kolonne blir resultatet av disse `NULL`
- ◆ `count(*)` teller med `NULL`-verdier
- ◆ Men dersom vi oppgir en konkret kolonne, f.eks. `count(product_name)` vil den kun telle verdiene som ikke er `NULL`
- ◆ For eksempel:

Name	Age
Per	2
Kari	4
Mari	NULL

```
SELECT min(Age) FROM Person;    --> 2
SELECT avg(Age) FROM Person;    --> 3
SELECT count(Age) FROM Person;  --> 2
SELECT count(*) FROM Person;
```

```
SELECT sum(Age) FROM Person
WHERE Name = 'Mari';
```

```
SELECT count(Age) FROM Person
WHERE Name = 'Mari';
```

# Aggregering og NULL

- ◆ Aggregering med `sum`, `min`, `max` og `avg` ignorerer `NULL`-verdier
- ◆ Det betyr også at dersom det kun er `NULL`-verdier i en kolonne blir resultatet av disse `NULL`
- ◆ `count(*)` teller med `NULL`-verdier
- ◆ Men dersom vi oppgir en konkret kolonne, f.eks. `count(product_name)` vil den kun telle verdiene som ikke er `NULL`
- ◆ For eksempel:

Name	Age
Per	2
Kari	4
Mari	NULL

```
SELECT min(Age) FROM Person;    --> 2
SELECT avg(Age) FROM Person;    --> 3
SELECT count(Age) FROM Person;  --> 2
SELECT count(*) FROM Person;    --> 3
```

```
SELECT sum(Age) FROM Person
WHERE Name = 'Mari';
```

```
SELECT count(Age) FROM Person
WHERE Name = 'Mari';
```

# Aggregering og NULL

- ◆ Aggregering med `sum`, `min`, `max` og `avg` ignorerer `NULL`-verdier
- ◆ Det betyr også at dersom det kun er `NULL`-verdier i en kolonne blir resultatet av disse `NULL`
- ◆ `count(*)` teller med `NULL`-verdier
- ◆ Men dersom vi oppgir en konkret kolonne, f.eks. `count(product_name)` vil den kun telle verdiene som ikke er `NULL`
- ◆ For eksempel:

Name	Age
Per	2
Kari	4
Mari	NULL

```
SELECT min(Age) FROM Person;    --> 2
SELECT avg(Age) FROM Person;    --> 3
SELECT count(Age) FROM Person;  --> 2
SELECT count(*) FROM Person;    --> 3
```

```
SELECT sum(Age) FROM Person
WHERE Name = 'Mari';            --> NULL
```

```
SELECT count(Age) FROM Person
WHERE Name = 'Mari';
```

# Aggregering og NULL

- ◆ Aggregering med `sum`, `min`, `max` og `avg` ignorerer `NULL`-verdier
- ◆ Det betyr også at dersom det kun er `NULL`-verdier i en kolonne blir resultatet av disse `NULL`
- ◆ `count(*)` teller med `NULL`-verdier
- ◆ Men dersom vi oppgir en konkret kolonne, f.eks. `count(product_name)` vil den kun telle verdiene som ikke er `NULL`
- ◆ For eksempel:

Name	Age
Per	2
Kari	4
Mari	NULL

```
SELECT min(Age) FROM Person;    --> 2
SELECT avg(Age) FROM Person;    --> 3
SELECT count(Age) FROM Person;  --> 2
SELECT count(*) FROM Person;    --> 3
```

```
SELECT sum(Age) FROM Person
WHERE Name = 'Mari';           --> NULL
```

```
SELECT count(Age) FROM Person
WHERE Name = 'Mari';          --> 0
```

# Repetisjon: Inner joins

---

Hvilken kunde har kjøpt hvilket produkt?

```
SELECT ProductName, Customer
FROM products AS p INNER JOIN orders AS o
ON p.ProductID = o.ProductID
```

Resultat

# Repetisjon: Inner joins

Hvilken kunde har kjøpt hvilket produkt?

```
SELECT ProductName, Customer
FROM products AS p INNER JOIN orders AS o
ON p.ProductID = o.ProductID
```

Resultat

products		
ProductID	Name	Price
0	TV 50 inch	8999
1	Laptop 2.5GHz	7499

orders		
OrderID	ProductID	Customer
0	1	John Mill
1	1	Peter Smith
2	0	Anna Consuma
3	1	Yvonne Potter

# Repetisjon: Inner joins

Hvilken kunde har kjøpt hvilket produkt?

```
SELECT ProductName, Customer
FROM products AS p INNER JOIN orders AS o
ON p.ProductID = o.ProductID
```

## Resultat

p.ProductID	p.ProductName	p.Price	o.OrderID	o.ProductID	o.Customer
0	TV 50 inch	8999	0	1	John Mill
0	TV 50 inch	8999	1	1	Peter Smith
0	TV 50 inch	8999	2	0	Anna Consuma
0	TV 50 inch	8999	3	1	Yvonne Potter
1	Laptop 2.5GHz	7499	0	1	John Mill
1	Laptop 2.5GHz	7499	1	1	Peter Smith
1	Laptop 2.5GHz	7499	2	0	Anna Consuma
1	Laptop 2.5GHz	7499	3	1	Yvonne Potter

# Repetisjon: Inner joins

Hvilken kunde har kjøpt hvilket produkt?

```
SELECT ProductName, Customer
FROM products AS p INNER JOIN orders AS o
ON p.ProductID = o.ProductID
```

## Resultat

p.ProductID	p.ProductName	p.Price	o.OrderID	o.ProductID	o.Customer
0	TV 50 inch	8999	0	1	John Mill
0	TV 50 inch	8999	1	1	Peter Smith
0	TV 50 inch	8999	2	0	Anna Consuma
0	TV 50 inch	8999	3	1	Yvonne Potter
1	Laptop 2.5GHz	7499	0	1	John Mill
1	Laptop 2.5GHz	7499	1	1	Peter Smith
1	Laptop 2.5GHz	7499	2	0	Anna Consuma
1	Laptop 2.5GHz	7499	3	1	Yvonne Potter



# Repetisjon: Inner joins

Hvilken kunde har kjøpt hvilket produkt?

```
SELECT ProductName, Customer
FROM products AS p INNER JOIN orders AS o
ON p.ProductID = o.ProductID
```

## Resultat

p.ProductID	p.ProductName	p.Price	o.OrderID	o.ProductID	o.Customer
0	TV 50 inch	8999	0	1	John Mill
0	TV 50 inch	8999	1	1	Peter Smith
0	TV 50 inch	8999	2	0	Anna Consuma
0	TV 50 inch	8999	3	1	Yvonne Potter
1	Laptop 2.5GHz	7499	0	1	John Mill
1	Laptop 2.5GHz	7499	1	1	Peter Smith
1	Laptop 2.5GHz	7499	2	0	Anna Consuma
1	Laptop 2.5GHz	7499	3	1	Yvonne Potter

# Repetisjon: Inner joins

Hvilken kunde har kjøpt hvilket produkt?

```
SELECT ProductName, Customer
FROM products AS p INNER JOIN orders AS o
ON p.ProductID = o.ProductID
```

Resultat

p.ProductName	o.Customer
TV 50 inch	Anna Consuma
Laptop 2.5GHz	John Mill
Laptop 2.5GHz	Peter Smith
Laptop 2.5GHz	Yvonne Potter

# Inner joins og manglende verdier

---

Hvilken kunde har kjøpt hvilket produkt?

```
SELECT ProductName, Customer
FROM products AS p INNER JOIN orders AS o
ON p.ProductID = o.ProductID
```

Resultat

# Inner joins og manglende verdier

Hvilken kunde har kjøpt hvilket produkt?

```
SELECT ProductName, Customer
FROM products AS p INNER JOIN orders AS o
ON p.ProductID = o.ProductID
```

Resultat

products		
ProductID	Name	Price
0	TV 50 inch	8999
1	Laptop 2.5GHz	7499
2	Noise-amplifying Headphones	9999

orders		
OrderID	ProductID	Customer
0	1	John Mill
1	1	Peter Smith
2	0	Anna Consuma
3	1	Yvonne Potter

# Inner joins og manglende verdier

Hvilken kunde har kjøpt hvilket produkt?

```
SELECT ProductName, Customer
FROM products AS p INNER JOIN orders AS o
ON p.ProductID = o.ProductID
```

## Resultat

p.ProductID	p.ProductName	p.Price	o.OrderID	o.ProductID	o.Customer
0	TV 50 inch	8999	0	1	John Mill
0	TV 50 inch	8999	1	1	Peter Smith
0	TV 50 inch	8999	2	0	Anna Consuma
0	TV 50 inch	8999	3	1	Yvonne Potter
1	Laptop 2.5GHz	7499	0	1	John Mill
1	Laptop 2.5GHz	7499	1	1	Peter Smith
1	Laptop 2.5GHz	7499	2	0	Anna Consuma
1	Laptop 2.5GHz	7499	3	1	Yvonne Potter
2	Noise-amplifying Headphones	9999	0	1	John Mill
2	Noise-amplifying Headphones	9999	1	1	Peter Smith
2	Noise-amplifying Headphones	9999	2	0	Anna Consuma
2	Noise-amplifying Headphones	9999	3	1	Yvonne Potter

# Inner joins og manglende verdier

Hvilken kunde har kjøpt hvilket produkt?

```
SELECT ProductName, Customer
FROM products AS p INNER JOIN orders AS o
ON p.ProductID = o.ProductID
```

## Resultat

p.ProductID	p.ProductName	p.Price	o.OrderID	o.ProductID	o.Customer
0	TV 50 inch	8999	0	1	John Mill
0	TV 50 inch	8999	1	1	Peter Smith
0	TV 50 inch	8999	2	0	Anna Consuma
0	TV 50 inch	8999	3	1	Yvonne Potter
1	Laptop 2.5GHz	7499	0	1	John Mill
1	Laptop 2.5GHz	7499	1	1	Peter Smith
1	Laptop 2.5GHz	7499	2	0	Anna Consuma
1	Laptop 2.5GHz	7499	3	1	Yvonne Potter
2	Noise-amplifying Headphones	9999	0	1	John Mill
2	Noise-amplifying Headphones	9999	1	1	Peter Smith
2	Noise-amplifying Headphones	9999	2	0	Anna Consuma
2	Noise-amplifying Headphones	9999	3	1	Yvonne Potter

# Inner joins og manglende verdier

Hvilken kunde har kjøpt hvilket produkt?

```
SELECT ProductName, Customer
FROM products AS p INNER JOIN orders AS o
ON p.ProductID = o.ProductID
```

## Resultat

p.ProductID	p.ProductName	p.Price	o.OrderID	o.ProductID	o.Customer
0	TV 50 inch	8999	0	1	John Mill
0	TV 50 inch	8999	1	1	Peter Smith
0	TV 50 inch	8999	2	0	Anna Consuma
0	TV 50 inch	8999	3	1	Yvonne Potter
1	Laptop 2.5GHz	7499	0	1	John Mill
1	Laptop 2.5GHz	7499	1	1	Peter Smith
1	Laptop 2.5GHz	7499	2	0	Anna Consuma
1	Laptop 2.5GHz	7499	3	1	Yvonne Potter
2	Noise-amplifying Headphones	9999	0	1	John Mill
2	Noise-amplifying Headphones	9999	1	1	Peter Smith
2	Noise-amplifying Headphones	9999	2	0	Anna Consuma
2	Noise-amplifying Headphones	9999	3	1	Yvonne Potter

# Inner joins og manglende verdier

Hvilken kunde har kjøpt hvilket produkt?

```
SELECT ProductName, Customer
FROM products AS p INNER JOIN orders AS o
ON p.ProductID = o.ProductID
```

Resultat

p.ProductName	o.Customer
TV 50 inch	Anna Consuma
Laptop 2.5GHz	John Mill
Laptop 2.5GHz	Peter Smith
Laptop 2.5GHz	Yvonne Potter



# Inner joins og manglende verdier med aggregater

---

Hvor mange har kjøpt hvert produkt?

```
SELECT p.ProductName, count(o.Customer) AS num
FROM products AS p INNER JOIN orders AS o
    ON p.ProductID = o.ProductID
GROUP BY p.ProductName
```

Resultat

# Inner joins og manglende verdier med aggregater

## Hvor mange har kjøpt hvert produkt?

```
SELECT p.ProductName, count(o.Customer) AS num
FROM products AS p INNER JOIN orders AS o
    ON p.ProductID = o.ProductID
GROUP BY p.ProductName
```

## Resultat

products		
ProductID	Name	Price
0	TV 50 inch	8999
1	Laptop 2.5GHz	7499
2	Noise-amplifying Headphones	9999

orders		
OrderID	ProductID	Customer
0	1	John Mill
1	1	Peter Smith
2	0	Anna Consuma
3	1	Yvonne Potter

# Inner joins og manglende verdier med aggregater

## Hvor mange har kjøpt hvert produkt?

```
SELECT p.ProductName, count(o.Customer) AS num
FROM products AS p INNER JOIN orders AS o
    ON p.ProductID = o.ProductID
GROUP BY p.ProductName
```

## Resultat

p.ProductID	p.ProductName	p.Price	o.OrderID	o.ProductID	o.Customer
0	TV 50 inch	8999	0	1	John Mill
0	TV 50 inch	8999	1	1	Peter Smith
0	TV 50 inch	8999	2	0	Anna Consuma
0	TV 50 inch	8999	3	1	Yvonne Potter
1	Laptop 2.5GHz	7499	0	1	John Mill
1	Laptop 2.5GHz	7499	1	1	Peter Smith
1	Laptop 2.5GHz	7499	2	0	Anna Consuma
1	Laptop 2.5GHz	7499	3	1	Yvonne Potter
2	Noise-amplifying Headphones	9999	0	1	John Mill
2	Noise-amplifying Headphones	9999	1	1	Peter Smith
2	Noise-amplifying Headphones	9999	2	0	Anna Consuma
2	Noise-amplifying Headphones	9999	3	1	Yvonne Potter

# Inner joins og manglende verdier med aggregater

## Hvor mange har kjøpt hvert produkt?

```
SELECT p.ProductName, count(o.Customer) AS num
FROM products AS p INNER JOIN orders AS o
    ON p.ProductID = o.ProductID
GROUP BY p.ProductName
```

## Resultat

p.ProductID	p.ProductName	p.Price	o.OrderID	o.ProductID	o.Customer
0	TV 50 inch	8999	0	1	John Mill
0	TV 50 inch	8999	1	1	Peter Smith
0	TV 50 inch	8999	2	0	Anna Consuma
0	TV 50 inch	8999	3	1	Yvonne Potter
1	Laptop 2.5GHz	7499	0	1	John Mill
1	Laptop 2.5GHz	7499	1	1	Peter Smith
1	Laptop 2.5GHz	7499	2	0	Anna Consuma
1	Laptop 2.5GHz	7499	3	1	Yvonne Potter
2	Noise-amplifying Headphones	9999	0	1	John Mill
2	Noise-amplifying Headphones	9999	1	1	Peter Smith
2	Noise-amplifying Headphones	9999	2	0	Anna Consuma
2	Noise-amplifying Headphones	9999	3	1	Yvonne Potter

# Inner joins og manglende verdier med aggregater

## Hvor mange har kjøpt hvert produkt?

```
SELECT p.ProductName, count(o.Customer) AS num
FROM products AS p INNER JOIN orders AS o
    ON p.ProductID = o.ProductID
GROUP BY p.ProductName
```

## Resultat

p.ProductName	num
TV 50 inch	1
Laptop 2.5GHz	3

# Problemer med Indre joins

---

- ◆ I forige spørring fikk vi ikke opp at 0 kunder har kjøpt Noise-amplifying Headset

## Problemer med Indre joins

---

- ◆ I forige spørring fikk vi ikke opp at 0 kunder har kjøpt Noise-amplifying Headset
- ◆ Årsaken er at den ikke joiner med noe, og derfor forsvinner fra svaret

## Problemer med Indre joins

---

- ◆ I forige spørring fikk vi ikke opp at 0 kunder har kjøpt Noise-amplifying Headset
- ◆ Årsaken er at den ikke joiner med noe, og derfor forsvinner fra svaret
- ◆ For å få ønsket resultat trenger vi altså en ny type join



# Problemer med Indre joins

---

- ◆ I forige spørring fikk vi ikke opp at 0 kunder har kjøpt Noise-amplifying Headset
- ◆ Årsaken er at den ikke joiner med noe, og derfor forsvinner fra svaret
- ◆ For å få ønsket resultat trenger vi altså en ny type join
- ◆ De nye joinene som løser problemet vårt heter ytre joins, eller *outer join* på engelsk

# Outer Joins

---

- ◆ Vi har flere varianter av ytre joins, nemlig
  - ◆ `left outer join`
  - ◆ `right outer join`
  - ◆ `full outer join`

# Outer Joins

---

- ◆ Vi har flere varianter av ytre joins, nemlig
  - ◆ `left outer join`
  - ◆ `right outer join`
  - ◆ `full outer join`
- ◆ Brukes ved å bytte ut `INNER JOIN` med f.eks. `LEFT OUTER JOIN`

# Outer Joins

---

- ◆ Vi har flere varianter av ytre joins, nemlig
  - ◆ `left outer join`
  - ◆ `right outer join`
  - ◆ `full outer join`
- ◆ Brukes ved å bytte ut `INNER JOIN` med f.eks. `LEFT OUTER JOIN`
- ◆ Hovedidéen bak denne typen join er å bevare alle rader fra en eller begge tabellene i joinen

# Outer Joins

---

- ◆ Vi har flere varianter av ytre joins, nemlig
  - ◆ `left outer join`
  - ◆ `right outer join`
  - ◆ `full outer join`
- ◆ Brukes ved å bytte ut `INNER JOIN` med f.eks. `LEFT OUTER JOIN`
- ◆ Hovedidéen bak denne typen join er å bevare alle rader fra en eller begge tabellene i joinen
- ◆ Og så fylle inn med `NULL` hvor vi ikke har noen match

# Left Outer Join

---

- ◆ I en *left outer join* vil alle rader i den venstre tabellen bli med i svaret

# Left Outer Join

---

- ◆ I en *left outer join* vil alle rader i den venstre tabellen bli med i svaret
- ◆ Resultatet av a `LEFT OUTER JOIN` b `ON` (a.c1 = b.c2) blir
  - ◆ samme som a `INNER JOIN` b `ON` (a.c1 = b.c2),

# Left Outer Join

---

- ◆ I en *left outer join* vil alle rader i den venstre tabellen bli med i svaret
- ◆ Resultatet av a `LEFT OUTER JOIN` b `ON` (a.c1 = b.c2) blir
  - ◆ samme som a `INNER JOIN` b `ON` (a.c1 = b.c2),
  - ◆ men hvor alle rader fra a som ikke matcher noen i b



# Left Outer Join

---

- ◆ I en *left outer join* vil alle rader i den venstre tabellen bli med i svaret
- ◆ Resultatet av a `LEFT OUTER JOIN` b `ON` (a.c1 = b.c2) blir
  - ◆ samme som a `INNER JOIN` b `ON` (a.c1 = b.c2),
  - ◆ men hvor alle rader fra a som ikke matcher noen i b
  - ◆ (altså hvor a.c1 ikke er lik noen b.c2)

# Left Outer Join

---

- ◆ I en *left outer join* vil alle rader i den venstre tabellen bli med i svaret
- ◆ Resultatet av a `LEFT OUTER JOIN` b `ON` (a.c1 = b.c2) blir
  - ◆ samme som a `INNER JOIN` b `ON` (a.c1 = b.c2),
  - ◆ men hvor alle rader fra a som ikke matcher noen i b
  - ◆ (altså hvor a.c1 ikke er lik noen b.c2)
  - ◆ blir lagt til resultatet, med `NULL` for alle bs kolonner

# Eksempel: Left Outer Join

---

## Left outer join mellom products og orders

```
SELECT *  
FROM products AS p LEFT OUTER JOIN orders AS o  
ON p.ProductID = o.ProductID;
```

## Resultat

# Eksempel: Left Outer Join

## Left outer join mellom products og orders

```
SELECT *  
FROM products AS p LEFT OUTER JOIN orders AS o  
ON p.ProductID = o.ProductID;
```

## Resultat

products		
ProductID	Name	Price
0	TV 50 inch	8999
1	Laptop 2.5GHz	7499
2	Noise-amplifying Headphones	9999

orders		
OrderID	ProductID	Customer
0	1	John Mill
1	1	Peter Smith
2	0	Anna Consuma
3	1	Yvonne Potter

# Eksempel: Left Outer Join

## Left outer join mellom products og orders

```
SELECT *
FROM products AS p LEFT OUTER JOIN orders AS o
ON p.ProductID = o.ProductID;
```

## Resultat

p.ProductID	p.ProductName	p.Price	o.OrderID	o.ProductID	o.Customer
0	TV 50 inch	8999	0	1	John Mill
0	TV 50 inch	8999	1	1	Peter Smith
0	TV 50 inch	8999	2	0	Anna Consuma
0	TV 50 inch	8999	3	1	Yvonne Potter
1	Laptop 2.5GHz	7499	0	1	John Mill
1	Laptop 2.5GHz	7499	1	1	Peter Smith
1	Laptop 2.5GHz	7499	2	0	Anna Consuma
1	Laptop 2.5GHz	7499	3	1	Yvonne Potter
2	Noise-amplifying Headphones	9999	0	1	John Mill
2	Noise-amplifying Headphones	9999	1	1	Peter Smith
2	Noise-amplifying Headphones	9999	2	0	Anna Consuma
2	Noise-amplifying Headphones	9999	3	1	Yvonne Potter

# Eksempel: Left Outer Join

## Left outer join mellom products og orders

```
SELECT *
FROM products AS p LEFT OUTER JOIN orders AS o
ON p.ProductID = o.ProductID;
```

## Resultat

p.ProductID	p.ProductName	p.Price	o.OrderID	o.ProductID	o.Customer
0	TV 50 inch	8999	0	1	John Mill
0	TV 50 inch	8999	1	1	Peter Smith
0	TV 50 inch	8999	2	0	Anna Consuma
0	TV 50 inch	8999	3	1	Yvonne Potter
1	Laptop 2.5GHz	7499	0	1	John Mill
1	Laptop 2.5GHz	7499	1	1	Peter Smith
1	Laptop 2.5GHz	7499	2	0	Anna Consuma
1	Laptop 2.5GHz	7499	3	1	Yvonne Potter
2	Noise-amplifying Headphones	9999	0	1	John Mill
2	Noise-amplifying Headphones	9999	1	1	Peter Smith
2	Noise-amplifying Headphones	9999	2	0	Anna Consuma
2	Noise-amplifying Headphones	9999	3	1	Yvonne Potter

# Eksempel: Left Outer Join

## Left outer join mellom products og orders

```
SELECT *  
FROM products AS p LEFT OUTER JOIN orders AS o  
ON p.ProductID = o.ProductID;
```

## Resultat

p.ProductID	p.ProductName	p.Price	o.OrderID	o.ProductID	o.Customer
0	TV 50 inch	8999	2	0	Anna Consuma
1	Laptop 2.5GHz	7499	0	1	John Mill
1	Laptop 2.5GHz	7499	1	1	Peter Smith
1	Laptop 2.5GHz	7499	3	1	Yvonne Potter

# Eksempel: Left Outer Join

## Left outer join mellom products og orders

```
SELECT *  
FROM products AS p LEFT OUTER JOIN orders AS o  
ON p.ProductID = o.ProductID;
```

## Resultat

p.ProductID	p.ProductName	p.Price	o.OrderID	o.ProductID	o.Customer
0	TV 50 inch	8999	2	0	Anna Consuma
1	Laptop 2.5GHz	7499	0	1	John Mill
1	Laptop 2.5GHz	7499	1	1	Peter Smith
1	Laptop 2.5GHz	7499	3	1	Yvonne Potter
2	Noise-amplifying Headphones	9999			



# Eksempel: Left Outer Join

## Left outer join mellom products og orders

```
SELECT *
FROM products AS p LEFT OUTER JOIN orders AS o
ON p.ProductID = o.ProductID;
```

## Resultat

p.ProductID	p.ProductName	p.Price	o.OrderID	o.ProductID	o.Customer
0	TV 50 inch	8999	2	0	Anna Consuma
1	Laptop 2.5GHz	7499	0	1	John Mill
1	Laptop 2.5GHz	7499	1	1	Peter Smith
1	Laptop 2.5GHz	7499	3	1	Yvonne Potter
2	Noise-amplifying Headphones	9999	NULL	NULL	NULL

# Eksempel: Left Outer Join

---

Hvor mange har kjøpt hvert produkt?

```
SELECT p.ProductName, count(o.Customer) AS num
FROM products AS p LEFT OUTER JOIN orders AS o
ON p.ProductID = o.ProductID
GROUP BY p.ProductName
```

Resultat

# Eksempel: Left Outer Join

## Hvor mange har kjøpt hvert produkt?

```
SELECT p.ProductName, count(o.Customer) AS num
FROM products AS p LEFT OUTER JOIN orders AS o
ON p.ProductID = o.ProductID
GROUP BY p.ProductName
```

## Resultat

products		
ProductID	Name	Price
0	TV 50 inch	8999
1	Laptop 2.5GHz	7499
2	Noise-amplifying Headphones	9999

orders		
OrderID	ProductID	Customer
0	1	John Mill
1	1	Peter Smith
2	0	Anna Consuma
3	1	Yvonne Potter

# Eksempel: Left Outer Join

Hvor mange har kjøpt hvert produkt?

```
SELECT p.ProductName, count(o.Customer) AS num
FROM products AS p LEFT OUTER JOIN orders AS o
ON p.ProductID = o.ProductID
GROUP BY p.ProductName
```

## Resultat

p.ProductID	p.ProductName	p.Price	o.OrderID	o.ProductID	o.Customer
0	TV 50 inch	8999	2	0	Anna Consuma
1	Laptop 2.5GHz	7499	0	1	John Mill
1	Laptop 2.5GHz	7499	1	1	Peter Smith
1	Laptop 2.5GHz	7499	3	1	Yvonne Potter

# Eksempel: Left Outer Join

Hvor mange har kjøpt hvert produkt?

```
SELECT p.ProductName, count(o.Customer) AS num
FROM products AS p LEFT OUTER JOIN orders AS o
ON p.ProductID = o.ProductID
GROUP BY p.ProductName
```

## Resultat

p.ProductID	p.ProductName	p.Price	o.OrderID	o.ProductID	o.Customer
0	TV 50 inch	8999	2	0	Anna Consuma
1	Laptop 2.5GHz	7499	0	1	John Mill
1	Laptop 2.5GHz	7499	1	1	Peter Smith
1	Laptop 2.5GHz	7499	3	1	Yvonne Potter
2	Noise-amplifying Headphones	9999	NULL	NULL	NULL

# Eksempel: Left Outer Join

## Hvor mange har kjøpt hvert produkt?

```
SELECT p.ProductName, count(o.Customer) AS num
FROM products AS p LEFT OUTER JOIN orders AS o
ON p.ProductID = o.ProductID
GROUP BY p.ProductName
```

## Resultat

p.ProductName	num
TV 50 inch	1
Laptop 2.5GHz	3
Noise-amplifying Headphones	0

## Andre nyttige bruksområder for ytre joins

---

- ◆ Som vi ser er ytre joins nyttige når vi aggregerer, for å ikke miste resultater underveis

## Andre nyttige bruksområder for ytre joins

---

- ◆ Som vi ser er ytre joins nyttige når vi aggregerer, for å ikke miste resultater underveis
- ◆ Ytre joins kan også være nyttige for å kombinere ufullstendig informasjon fra flere tabeller



## Andre nyttige bruksområder for ytre joins

---

- ◆ Som vi ser er ytre joins nyttige når vi aggregerer, for å ikke miste resultater underveis
- ◆ Ytre joins kan også være nyttige for å kombinere ufullstendig informasjon fra flere tabeller
- ◆ For eksempel:

Persons

ID	Name
1	Per
2	Mari
3	Ida

Numbers

ID	Phone
1	48123456
3	98765432

Emails

ID	Email
1	per@mail.no
2	mari@umail.net

## Andre nyttige bruksområder for ytre joins

- ◆ Som vi ser er ytre joins nyttige når vi aggregerer, for å ikke miste resultater underveis
- ◆ Ytre joins kan også være nyttige for å kombinere ufullstendig informasjon fra flere tabeller
- ◆ For eksempel:

ID	Name
1	Per
2	Mari
3	Ida

ID	Phone
1	48123456
3	98765432

ID	Email
1	per@mail.no
2	mari@umail.net

```
SELECT p.Name, n.Phone, e.Email
FROM Persons AS p
LEFT OUTER JOIN Numbers AS n
ON (p.ID = n.ID)
LEFT OUTER JOIN Emails AS e
ON (p.ID = e.ID);
```

## Andre nyttige bruksområder for ytre joins

- ◆ Som vi ser er ytre joins nyttige når vi aggregerer, for å ikke miste resultater underveis
- ◆ Ytre joins kan også være nyttige for å kombinere ufullstendig informasjon fra flere tabeller
- ◆ For eksempel:

ID	Name
1	Per
2	Mari
3	Ida

ID	Phone
1	48123456
3	98765432

ID	Email
1	per@mail.no
2	mari@umail.net

```
SELECT p.Name, n.Phone, e.Email
FROM Persons AS p
     LEFT OUTER JOIN Numbers AS n
       ON (p.ID = n.ID)
     LEFT OUTER JOIN Emails AS e
       ON (p.ID = e.ID);
```

p.Name	n.Phone	e.Email
Per	48123456	per@mail.no
Mari	NULL	mari@umail.net
Ida	98765432	NULL

## Andre ytre joins

---

- ◆ a `RIGHT OUTER JOIN` b `ON` (a.c1 = b.c2) er akkurat det samme som  
b `LEFT OUTER JOIN` a `ON` (b.c2 = a.c1)

## Andre ytre joins

---

- ◆ a `RIGHT OUTER JOIN` b `ON` (a.c1 = b.c2) er akkurat det samme som b `LEFT OUTER JOIN` a `ON` (b.c2 = a.c1)
- ◆ Altså, i en *right outer join* vil alle radene i den høyre tabellen være med i resultatet

## Andre ytre joins

---

- ◆ a `RIGHT OUTER JOIN` b `ON` (a.c1 = b.c2) er akkurat det samme som b `LEFT OUTER JOIN` a `ON` (b.c2 = a.c1)
- ◆ Altså, i en *right outer join* vil alle radene i den høyre tabellen være med i resultatet
- ◆ Vi har også en `FULL OUTER JOIN` som er en slags kombinasjon, her vil ALLE rader være med i svaret

## Andre ytre joins

---

- ◆ a **RIGHT OUTER JOIN** b **ON** (a.c1 = b.c2) er akkurat det samme som b **LEFT OUTER JOIN** a **ON** (b.c2 = a.c1)
- ◆ Altså, i en *right outer join* vil alle radene i den høyre tabellen være med i resultatet
- ◆ Vi har også en **FULL OUTER JOIN** som er en slags kombinasjon, her vil ALLE rader være med i svaret
- ◆ For eksempel:

ID	Name
1	Per
2	Mari

ID	Phone
1	48123456
3	98765432

## Andre ytre joins

- ◆ a `RIGHT OUTER JOIN` b `ON` (a.c1 = b.c2) er akkurat det samme som b `LEFT OUTER JOIN` a `ON` (b.c2 = a.c1)
- ◆ Altså, i en *right outer join* vil alle radene i den høyre tabellen være med i resultatet
- ◆ Vi har også en `FULL OUTER JOIN` som er en slags kombinasjon, her vil ALLE rader være med i svaret
- ◆ For eksempel:

ID	Name
1	Per
2	Mari

ID	Phone
1	48123456
3	98765432

```
SELECT p.Name, n.Phone
FROM Persons AS p
FULL OUTER JOIN Numbers AS n
ON (p.ID = n.ID);
```



## Andre ytre joins

- ◆ a `RIGHT OUTER JOIN` b `ON` (a.c1 = b.c2) er akkurat det samme som b `LEFT OUTER JOIN` a `ON` (b.c2 = a.c1)
- ◆ Altså, i en *right outer join* vil alle radene i den høyre tabellen være med i resultatet
- ◆ Vi har også en `FULL OUTER JOIN` som er en slags kombinasjon, her vil ALLE rader være med i svaret
- ◆ For eksempel:

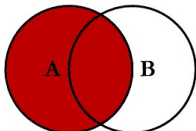
ID	Name
1	Per
2	Mari

ID	Phone
1	48123456
3	98765432

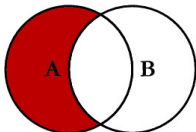
```
SELECT p.Name, n.Phone
FROM Persons AS p
FULL OUTER JOIN Numbers AS n
ON (p.ID = n.ID);
```

p.Name	n.Phone
Per	48123456
Mari	NULL
NULL	98765432

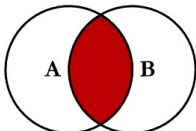
## SQL JOINS



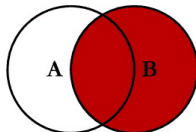
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key
```



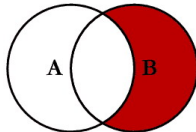
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key  
WHERE B.Key IS NULL.
```



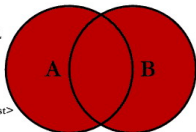
```
SELECT <select_list>  
FROM TableA A  
INNER JOIN TableB B  
ON A.Key = B.Key
```



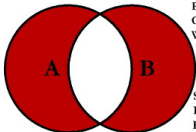
```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key
```



```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key  
WHERE A.Key IS NULL.
```



```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.Key = B.Key
```



```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.Key = B.Key  
WHERE A.Key IS NULL  
OR B.Key IS NULL.
```

## Ytre join-eksempel (1)

---

Finn navn på alle kunder som har gjort 2 eller færre bestillinger

## Ytre join-eksempel (1)

---

Finn navn på alle kunder som har gjort 2 eller færre bestillinger

```
SELECT c.company_name, count(o.order_id) AS num_orders
      FROM customers AS c
           LEFT OUTER JOIN orders AS o USING (customer_id)
GROUP BY c.company_name
HAVING count(o.order_id) <= 2;
```

## Ytre join-eksempel (2)

---

Finn ut for hvor mange produkter i hver kategori firmaet Leka Trading  
supplier

## Ytre join-eksempel (2)

Finn ut for hvor mange produkter i hver kategori firmaet Leka Trading  
supplier

```
WITH
  supplies AS (
    SELECT category_id
    FROM suppliers INNER JOIN products USING (supplier_id)
    WHERE company_name = 'Leka Trading'
  )
SELECT c.category_name, count(s.category_id) AS nr_products
FROM categories AS c
  LEFT OUTER JOIN supplies AS s USING (category_id)
GROUP BY c.category_name;
```

# Syntaks for joins

---

I stedet for

- ◆ `LEFT OUTER JOIN` kan man skrive `LEFT JOIN`

# Syntaks for joins

---

I stedet for

- ◆ `LEFT OUTER JOIN` kan man skrive `LEFT JOIN`
- ◆ `RIGHT OUTER JOIN` kan man skrive `RIGHT JOIN`



# Syntaks for joins

---

I stedet for

- ◆ `LEFT OUTER JOIN` kan man skrive `LEFT JOIN`
- ◆ `RIGHT OUTER JOIN` kan man skrive `RIGHT JOIN`
- ◆ `FULL OUTER JOIN` kan man skrive `FULL JOIN`

# Syntaks for joins

---

I stedet for

- ◆ `LEFT OUTER JOIN` kan man skrive `LEFT JOIN`
- ◆ `RIGHT OUTER JOIN` kan man skrive `RIGHT JOIN`
- ◆ `FULL OUTER JOIN` kan man skrive `FULL JOIN`
- ◆ `INNER JOIN` kan man skrive `JOIN`

Takk for nå!

---

Neste video handler om mengdeoperatorer.