



Databasteknik

JOIN, Views

Utbildare: Mikael Olsson

mikael.olsson@emmio.se

076-174 90 43

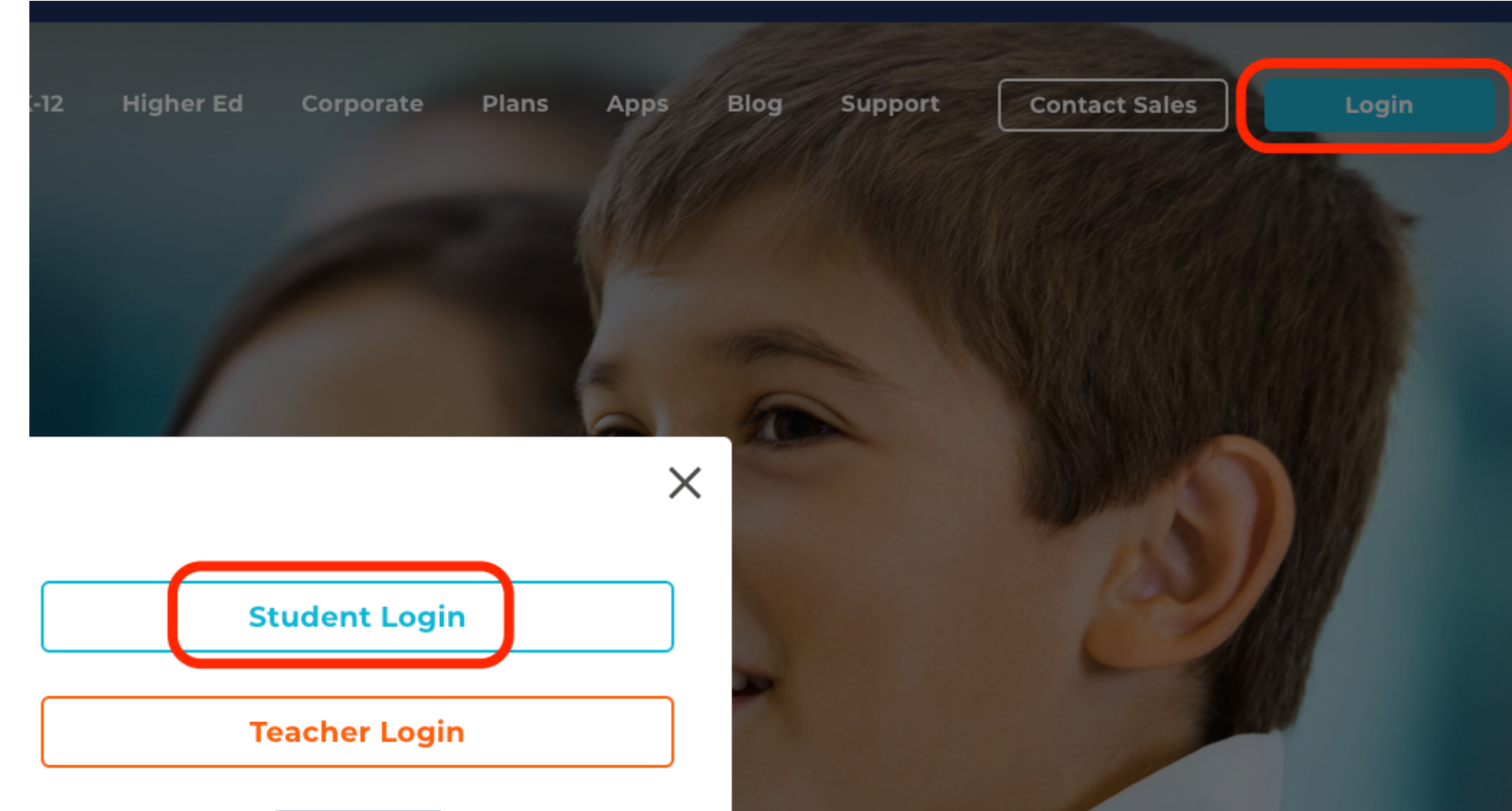
NACKADEMIN

Socrative

Quiz och snabba frågor

<https://www.socrative.com/>

- Logga in som student
- Ange rum “Emmio”
- Få upp en vänta-skärm
- Finns även som mobilapp:
<https://www.socrative.com/apps/>



Student Login

Room Name

JOIN



Waiting for the next activity to begin...

Socrative

Frågehanterare

- Jag kan ställa quiz eller snabbfrågor till klassen för att få en bättre bild över vad vi behöver öva mer på.
 - Multiple choice
 - Sant / falskt
 - Fritt svar
- När jag startar frågan / quizet kommer det automatiskt att dyka upp hos er.

Socratic

Multiple choice

- Vilka städer ligger i Sverige?
 - A. Helsingborg
 - B. Helsingfors
 - C. Kiruna
 - D. Oslo

Socrative

True / false

- Det är fredag idag.

Socrative

Fritt svar

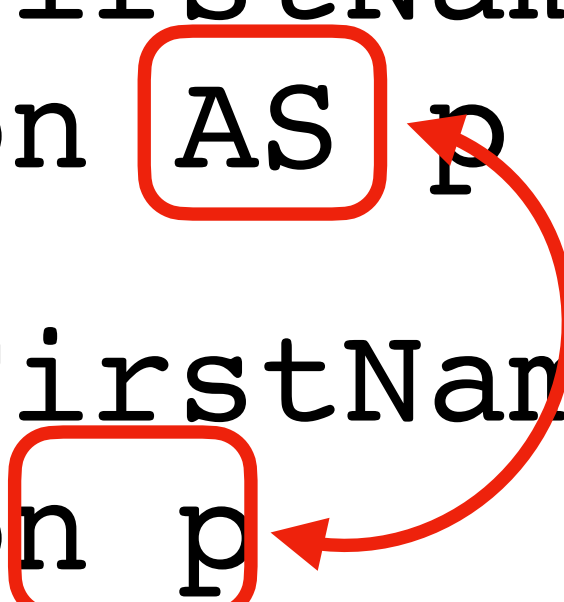
- Vad ska du göra i helgen?

Socrative

Ett första quiz

- Vi ska köra ett quiz med frågor baserade på första dagen.
- När jag startar quizet kommer det automatiskt att dyka upp hos er.
- Du som individ kan se detta som en hjälp. Vad kan du och vad behöver du träna mer på?
- Jag som lärare kan se om det är något särskilt område vi behöver träna mer på.

Alias

- Än så länge väljer vi bara data från en tabell, men när vi vill koppla ihop flera tabeller kommer vi att behöva ange vilken tabell våra fält kommer ifrån.
 - `SELECT Person.FirstName, Person.LastName
FROM Person`
 - För att göra det lite smidigare kan vi ge tabeller alias.
 - `SELECT p.FirstName, p.LastName
FROM Person AS p` 'AS' är frivillig
 - `SELECT p.FirstName, p.LastName
FROM Person p`
- 

Alias

- Alias kan även användas för att döpa om kolumner eller beräknade värden.

```
1 SELECT first_name AS contact_person
2 FROM person p
3
```

00% 36:1

Result Grid Filter Rows: Search Export:

contact_person	
Eva	
Sten	
Fredrik	
Stina	
Niklas	

```
1 SELECT CONCAT(first_name, ' ', last_name) AS name
2 FROM person
```

00% 50:1

Result Grid Filter Rows: Search Export:

name	
Eva Vik	
Sten Vik	
Fredrik Vik	
Stina Nilsson	
Niklas Nilsson	

JOIN

- Hur kopplar vi ihop tabeller?

id	first_name	last_name	address_id
1	Eva	Vik	1
2	Sten	Vik	1
3	Fredrik	Vik	1
4	Stina	Nilsson	2
5	Niklas	Nilsson	2

id	address	rooms
1	Vägen 1	3
2	Gatan 3	1

JOIN

- För att kombinera resultatet använder vi olika typer av JOIN.
- Den första typen vi ska titta på heter INNER JOIN.
- INNER är implicit. Om vi bara skriver JOIN kommer MySQL att tolka det som INNER JOIN.
- Andra typer är LEFT / RIGHT OUTER JOIN samt ett par relaterade varianter som vi återkommer till.

JOIN

INNER JOIN

- INNER JOIN kommer att ge oss alla möjliga kombinationer av de båda tabellerna, vilket initialt inte ser så användbart ut.
- Raderna har inget samband just nu.

```
SELECT *  
  FROM person p  
       JOIN address a
```

id	first_name	last_name	address_id	id	address	rooms
1	Eva	Vik	1	1	Vägen 1	3
1	Eva	Vik	1	2	Gatan 3	1
2	Sten	Vik	1	1	Vägen 1	3
2	Sten	Vik	1	2	Gatan 3	1
3	Fredrik	Vik	1	1	Vägen 1	3
3	Fredrik	Vik	1	2	Gatan 3	1
4	Stina	Nilsson	2	1	Vägen 1	3
4	Stina	Nilsson	2	2	Gatan 3	1
5	Niklas	Nilsson	2	1	Vägen 1	3
5	Niklas	Nilsson	2	2	Gatan 3	1

JOIN


INNER JOIN

- I INNER JOIN kan man ange ett eller flera villkor med hjälp av nyckelordet ON.
- ON fungerar ungefär som WHERE. Vi kommer bara att få med de rader som uppfyller villkoret.
- Vi kan sätta ett villkor om att jämföra värden i de olika tabellerna.

```
SELECT *  
FROM person p  
JOIN address a ON p.address_id = a.id
```

id	first_name	last_name	address_id	id	address	rooms
1	Eva	Vik	1	1	Vägen 1	3
2	Sten	Vik	1	1	Vägen 1	3
3	Fredrik	Vik	1	1	Vägen 1	3
4	Stina	Nilsson	2	2	Gatan 3	1
5	Niklas	Nilsson	2	2	Gatan 3	1

JOIN

- Vad händer om det finns rader i den ena tabellen som inte finns i den andra?
- Bor någon på Gränden 8? (True/False)  Socratic

id	first_name	last_name	address_id
1	Eva	Vik	1
2	Sten	Vik	1
3	Fredrik	Vik	1
4	Stina	Nilsson	2
5	Niklas	Nilsson	2
NULL	NULL	NULL	NULL

id	address	rooms
1	Vägen 1	3
2	Gatan 3	1
3	Gränden 8	4

JOIN

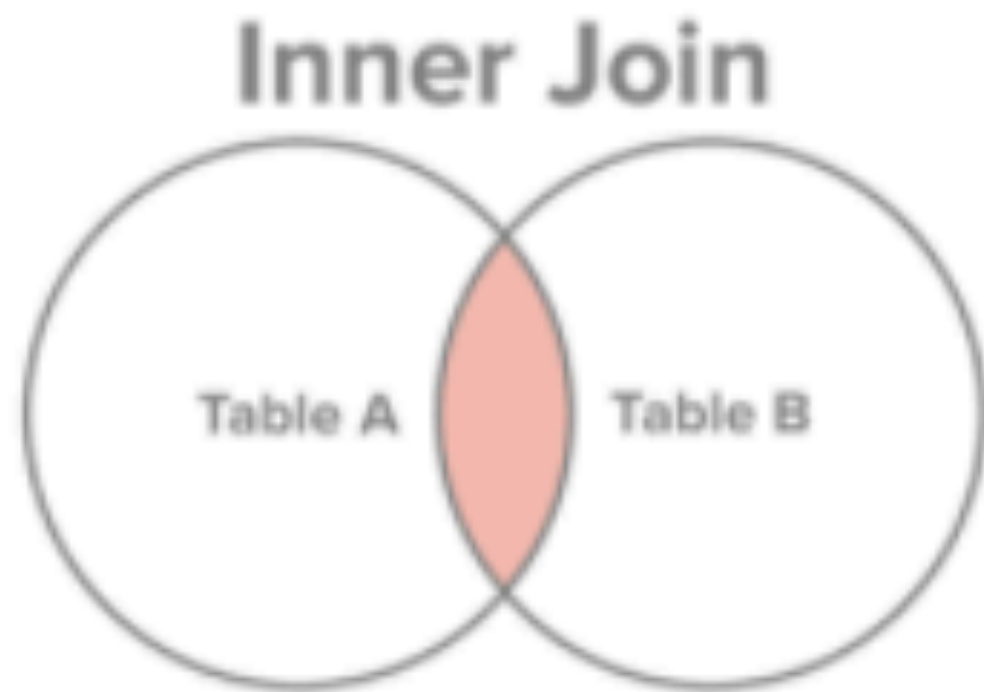
OUTER JOIN

- INNER JOIN ger oss de rader som finns i båda tabellerna.
- OUTER JOIN ger oss de rader som finns i båda tabellerna PLUS de rader som inte har någon motsvarighet i den ena tabellen.
- LEFT OUTER JOIN ger oss alla rader som finns i den vänstra tabellen (plus de matchande raderna i den högra tabellen).
- RIGHT OUTER JOIN ger oss alla rader som finns i den högra tabellen (plus de matchande raderna i den vänstra tabellen).
- Skriver man LEFT JOIN kommer MySQL att tolka det som LEFT OUTER JOIN och motsvarande för RIGHT.

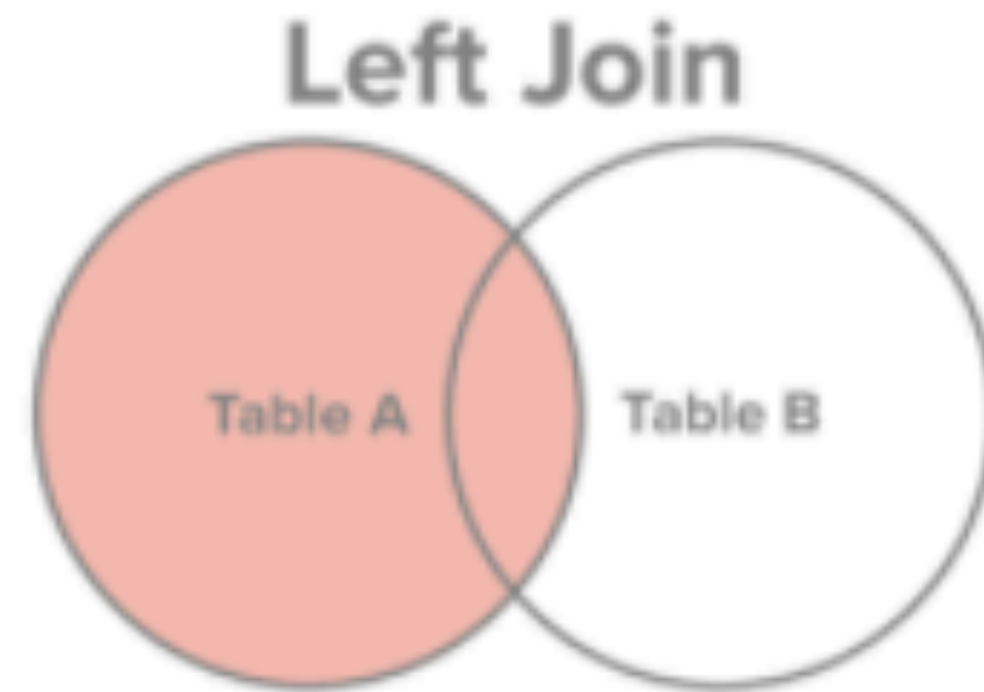
JOIN

OUTER JOIN

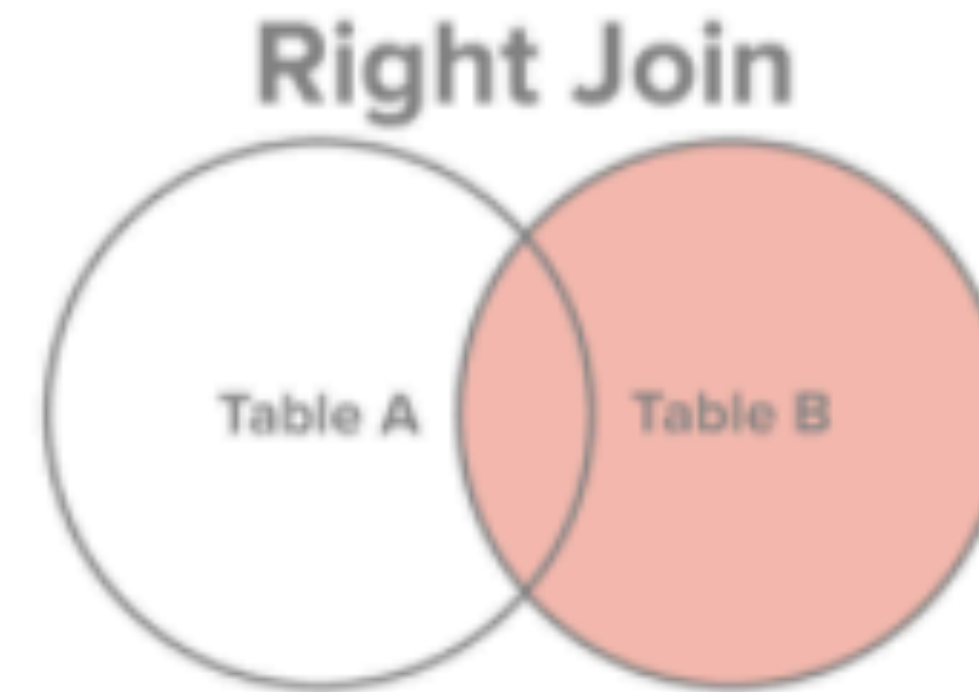
- MySQL har inte stöd för FULL JOIN.



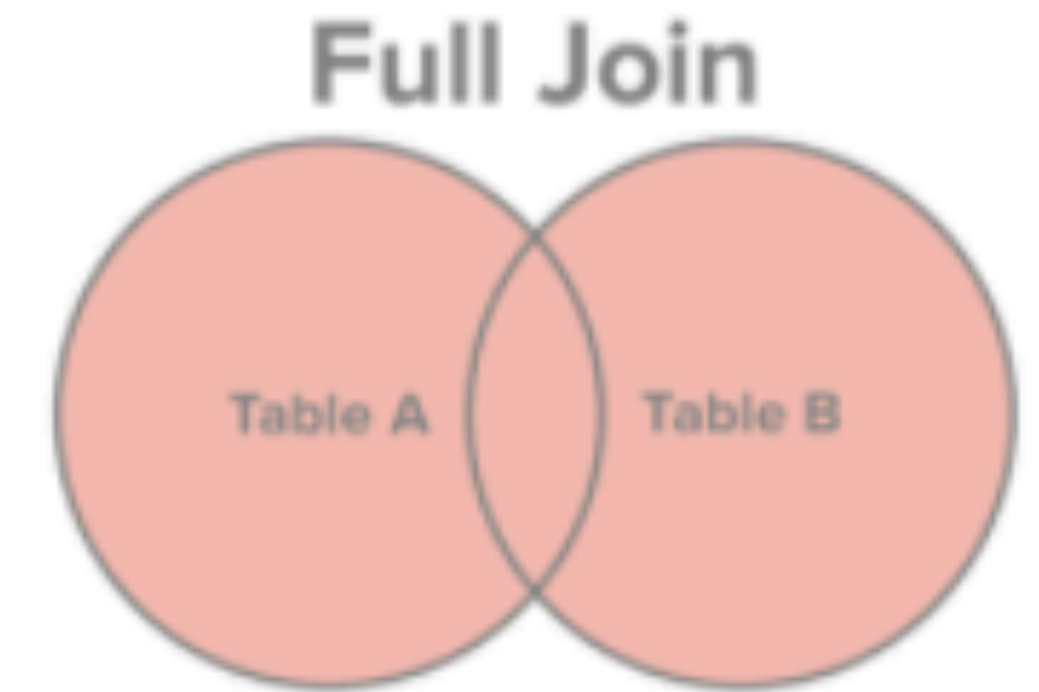
Select all records from Table A and Table B, where the join condition is met.



Select all records from Table A, along with records from Table B for which the join condition is met (if at all).



Select all records from Table B, along with records from Table A for which the join condition is met (if at all).



Select all records from Table A and Table B, regardless of whether the join condition is met or not.

OUTER JOIN

```
SELECT *
FROM person p
RIGHT JOIN address a ON p.address_id = a.id
```

id	first_name	last_name	address_id	id	address	rooms
1	Eva	Vik	1	1	Vägen 1	3
2	Sten	Vik	1	1	Vägen 1	3
3	Fredrik	Vik	1	1	Vägen 1	3
4	Stina	Nilsson	2	2	Gatan 3	1
5	Niklas	Nilsson	2	2	Gatan 3	1
NULL	NULL	NULL	NULL	3	Gränden 8	4

DISTINCT

- Tar bort alla dubletter *från resultatet*.

```
1 • SELECT *
2   FROM person p
3
```

100% 1:3

Result Grid Filter Rows: Search

	id	first_name	last_name	address_id
▶	1	Eva	Vik	1
	2	Sten	Vik	1
	3	Fredrik	Vik	1
	4	Stina	Nilsson	2
	5	Niklas	Nilsson	2

```
1 • SELECT DISTINCT *
2   FROM person p
3
```

100% 1:3

Result Grid Filter Rows: Search

	id	first_name	last_name	address_id
▶	1	Eva	Vik	1
	2	Sten	Vik	1
	3	Fredrik	Vik	1
	4	Stina	Nilsson	2
	5	Niklas	Nilsson	2

```
1 • SELECT last_name
2   FROM person p
3
```

100% 17:1

Result Grid Filter Rows: Search

	last_name
▶	Vik
	Vik
	Vik
	Nilsson
	Nilsson
	Nilsson

```
1 • SELECT DISTINCT last_name
2   FROM person p
3
```

100% 17:1

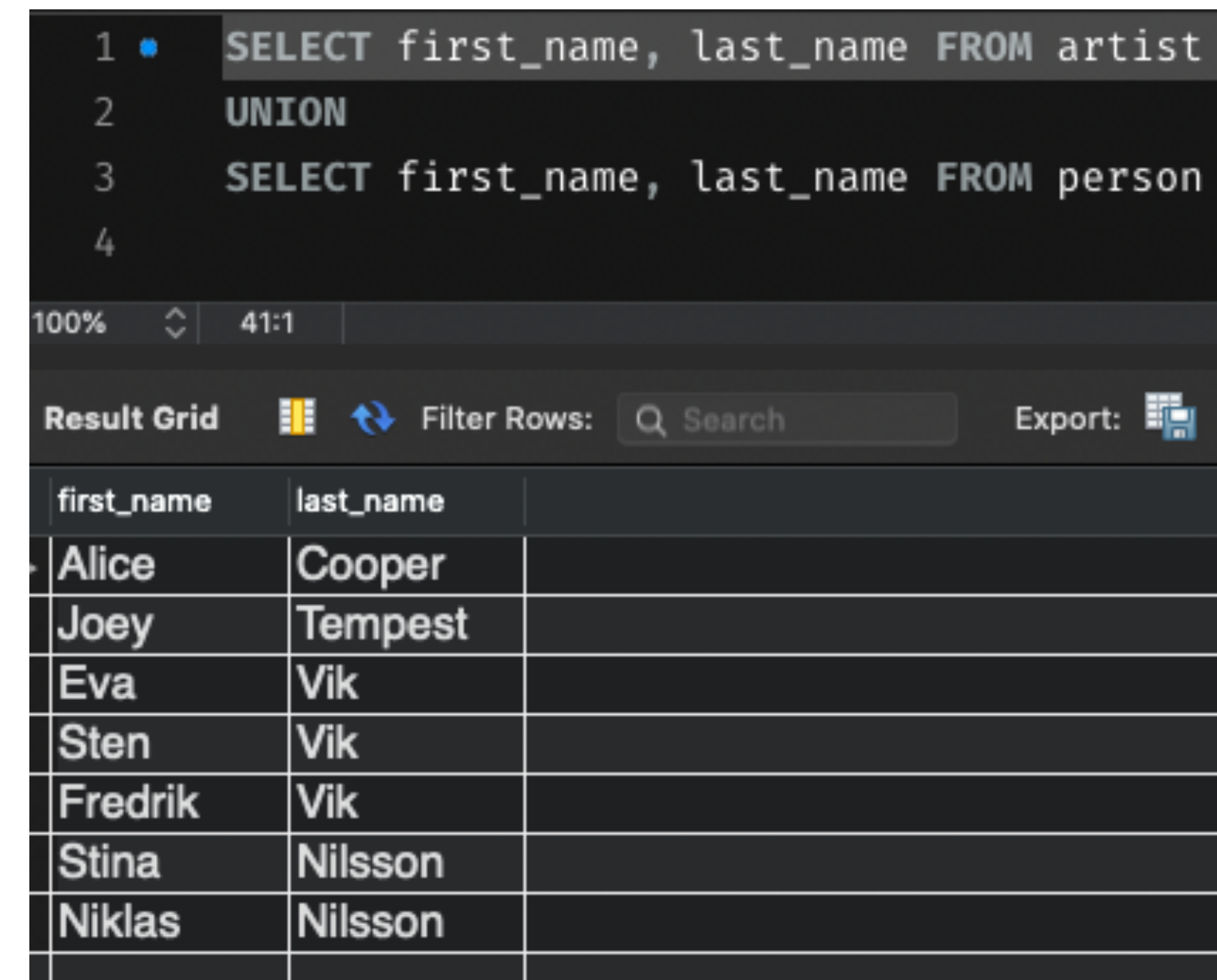
Result Grid Filter Rows: Search

	last_name
▶	Vik
	Nilsson

UNION

- JOIN låter oss slå ihop två tabeller "bredvid" varandra.
- UNION låter oss slå ihop två tabeller "efter" varandra.
- Frågorna måste ge lika många kolumner.
- UNION DISTINCT tar bort ev dubletter.

```
SELECT column_list  
UNION [DISTINCT | ALL]  
SELECT column_list
```



The screenshot shows a SQL query editor with a dark theme. The query is as follows:

```
1 SELECT first_name, last_name FROM artist  
2 UNION  
3 SELECT first_name, last_name FROM person  
4
```

Below the query editor, there is a toolbar with options for 'Result Grid', 'Filter Rows', and 'Export'. The 'Result Grid' is selected, and it displays the following data:

first_name	last_name
Alice	Cooper
Joey	Tempest
Eva	Vik
Sten	Vik
Fredrik	Vik
Stina	Nilsson
Niklas	Nilsson

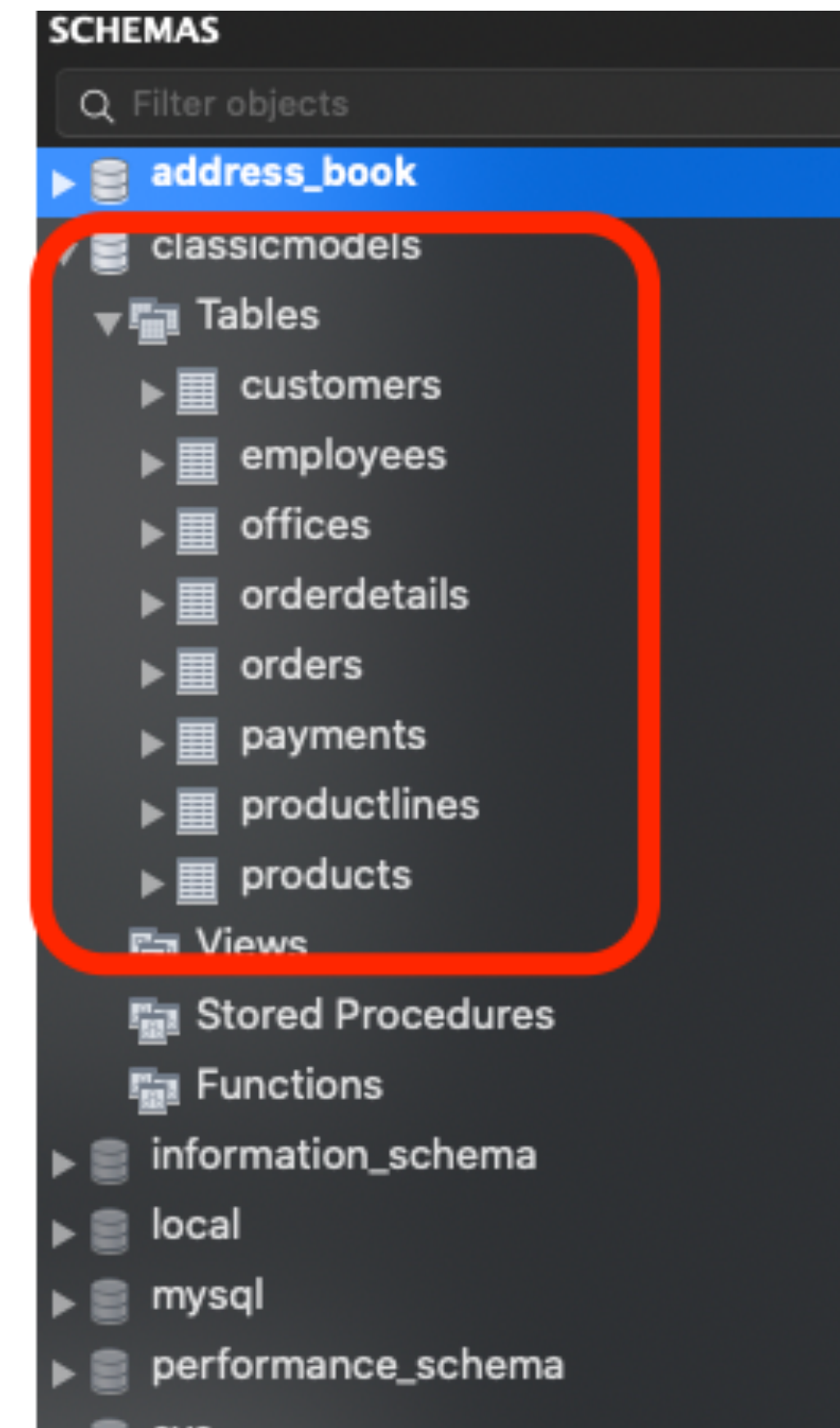
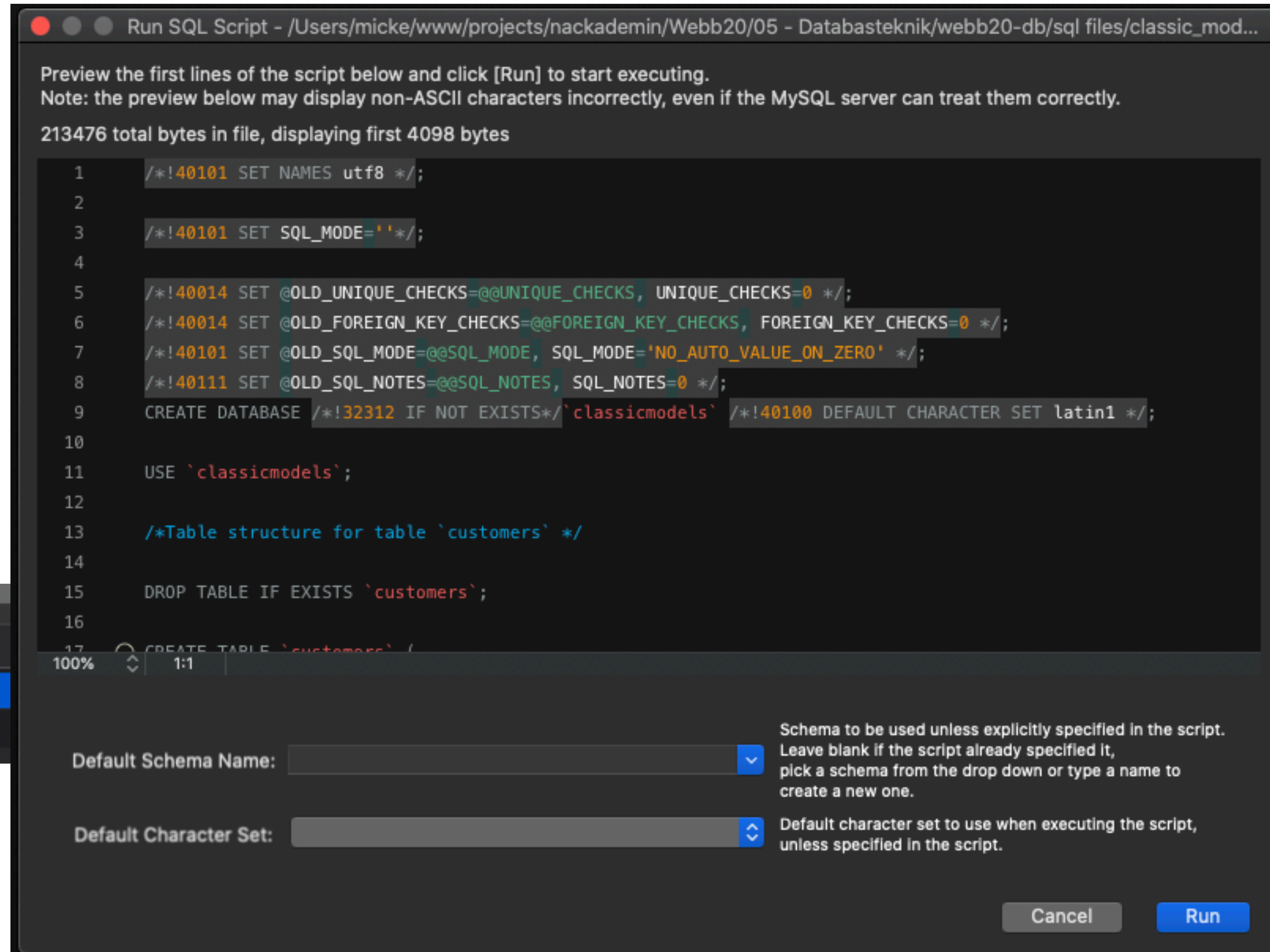
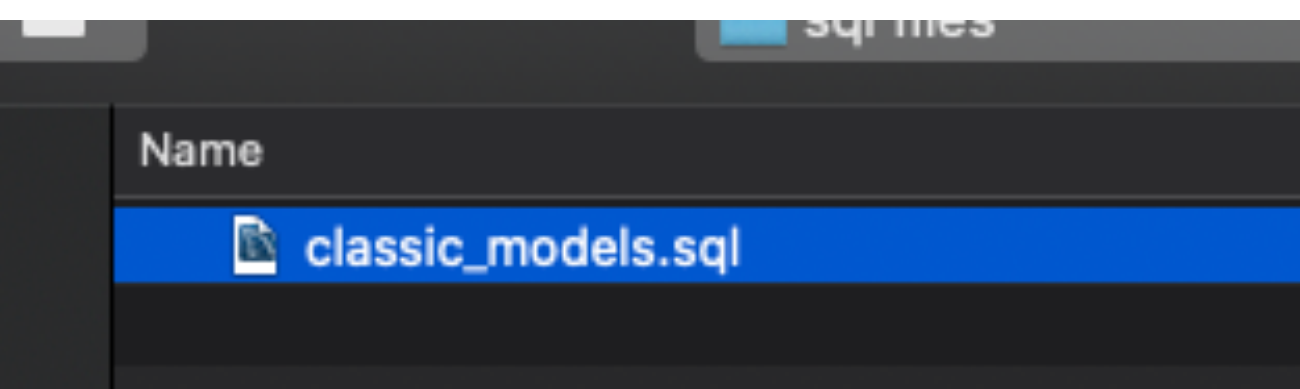
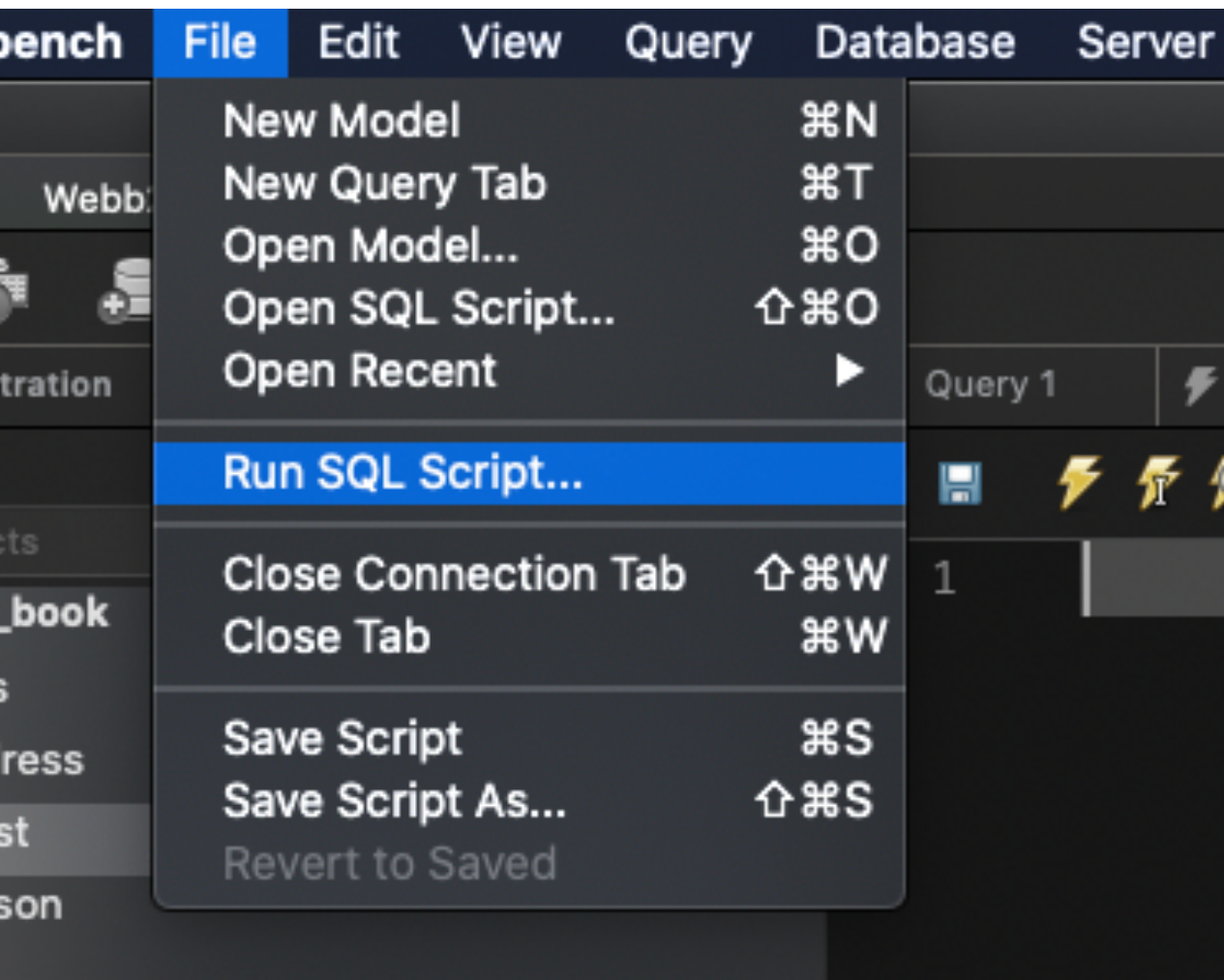
UNION, INTERSECT, MINUS

- Det finns fler sätt att kombinera tabeller. Vissa (t ex INTERSECT) finns inte inbyggt i MySQL, men det går att få till samma resultat.
- SQL Intersect, Union, Union All, Minus, and Except
<https://www.youtube.com/watch?v=bL5UX-p1wMc>

SQL i filer

- Allt vi gör i SQL / DDL görs med textkommandon.
- Det gör det lätt att exportera / importera databaser.
- Vi ska importera databasen classic_models. Den finns i repot under "sql files".

Classic Models



Övningar - Join

- Övningarna finns i repot tillsammans med lösningsförslag.
- Jobba tillsammans i grupper och gärna mellan grupperna om det behövs.

Views

- En View eller Vy är en virtuell tabell
- Kan t ex innehålla villkor och beräkningar

Views

- Säg att vi vill beräkna en summa för varje rad bestående av `quantityOrdered * priceEach`.
- Varför skulle det vara en dålig idé att spara den summan i en vanlig kolumn?

```
1 • SELECT * FROM classicmodels.orderdetails;
```

orderNumber	productCode	quantityOrdered	priceEach	orderLineNumber
10100	S18_1749	30	136.00	3
10100	S18_2248	50	55.09	2
10100	S18_4409	22	75.46	4
10100	S24_3969	49	35.29	1
10101	S18_2325	25	108.06	4
10101	S18_2795	26	167.06	1

Views

- Varför skulle det vara en dålig idé att spara summan i en vanlig kolumn?
- Därför att om summan inte skulle stämma, så vet vi inte vilket fält det beror på - vi har förstört tabellens integritet.

```
1 • SELECT * FROM classicmodels.orderdetails;
```

orderNumber	productCode	quantityOrdered	priceEach	orderLineNumber
10100	S18_1749	30	136.00	3
10100	S18_2248	50	55.09	2
10100	S18_4409	22	75.46	4
10100	S24_3969	49	35.29	1
10101	S18_2325	25	108.06	4
10101	S18_2795	26	167.06	1

Views

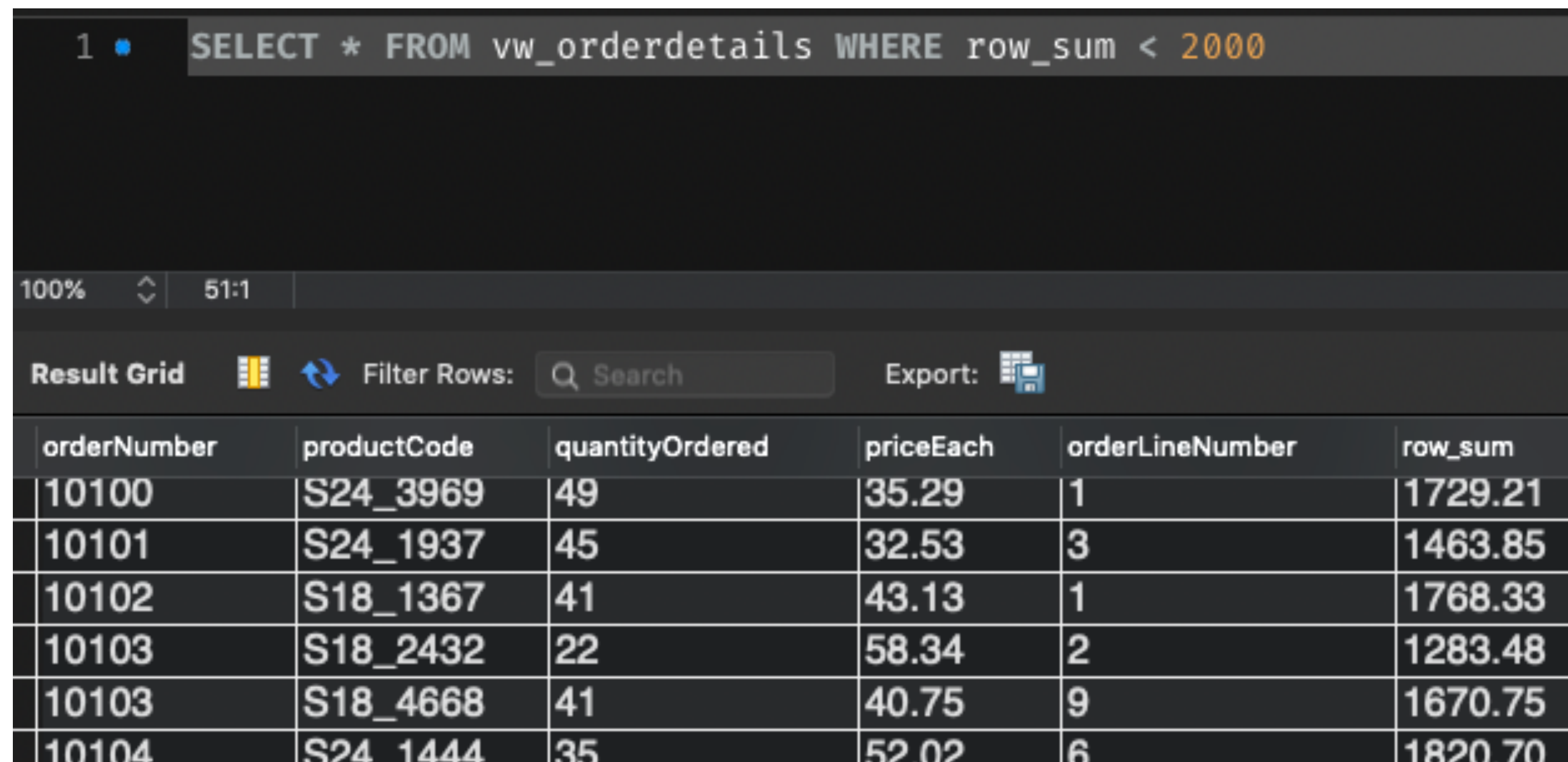
- Istället kan vi skapa en vy som innehåller ett beräknat fält.

```
CREATE VIEW `vw_orderdetails` AS  
  
SELECT *, quantityOrdered * priceEach AS row_sum FROM orderdetails
```

```
CREATE  
    ALGORITHM = UNDEFINED  
    DEFINER = `root`@`localhost`  
    SQL SECURITY DEFINER  
VIEW `vw_orderdetails` AS  
    SELECT  
        `orderdetails`.`orderNumber` AS `orderNumber`,  
        `orderdetails`.`productCode` AS `productCode`,  
        `orderdetails`.`quantityOrdered` AS `quantityOrdered`,  
        `orderdetails`.`priceEach` AS `priceEach`,  
        `orderdetails`.`orderLineNumber` AS `orderLineNumber`,  
        (`orderdetails`.`quantityOrdered` * `orderdetails`.`priceEach`) AS `row_sum`  
    FROM  
        `orderdetails`
```


Views

- En vy fungerar sedan likadant som en tabell.



```
1 • SELECT * FROM vw_orderdetails WHERE row_sum < 2000
```

100% 51:1

Result Grid Filter Rows: Search Export:

orderNumber	productCode	quantityOrdered	priceEach	orderLineNumber	row_sum
10100	S24_3969	49	35.29	1	1729.21
10101	S24_1937	45	32.53	3	1463.85
10102	S18_1367	41	43.13	1	1768.33
10103	S18_2432	22	58.34	2	1283.48
10103	S18_4668	41	40.75	9	1670.75
10104	S24_1444	35	52.02	6	1820.70

Views

Ett annat exempel

- Vi har ett gäng artiklar och vi vill inte att alla ska publiceras än av olika anledningar.
- På första sidan vill vi visa artiklar. Kommer "rätt" artiklar att visas med denna fråga?



Socrative

```
1 • SELECT * FROM article;
```

100% 15:1

Result Grid Filter Rows: Search

id	title	published
1	Lorem ipsum 1	1
2	Lorem ipsum 2	1
3	Lorem ipsum 3	0
NULL	NULL	NULL


Views

Ett annat exempel

- Varje gång vi vill hämta artiklar måste vi vara säkra på att lägga till ett villkor för att bara få med de artiklar som ska publiceras.
- Lätt att glömma!

```
1 • SELECT *
2   FROM article
3   WHERE published = 1;
```

00% 1:3

Result Grid   Filter Rows:

id	title	content	published
1	Lorem ipsum 1	Lorem ipsum 1	1
2	Lorem ipsum 2	Lorem ipsum 2	1
NULL	NULL	NULL	NULL

Views

Ett annat exempel

- Istället kan vi göra en vy som innehåller villkoret.
- Och använda vyn som en tabell.

```
CREATE
  ALGORITHM = UNDEFINED
  DEFINER = `root`@`localhost`
  SQL SECURITY DEFINER
VIEW `vw_published_articles` AS
  SELECT
    `article`.`id` AS `id`,
    `article`.`title` AS `title`,
    `article`.`content` AS `content`,
    `article`.`published` AS `published`
  FROM
    `article`
  WHERE
    (`article`.`published` = 1)
```

```
1 SELECT *
2 FROM vw_published_articles;
```

100% 6:2

Result Grid Filter Rows: Search Ex

id	title	content	published
1	Lorem ipsum 1	Lorem ipsum 1	1
2	Lorem ipsum 2	Lorem ipsum 2	1

Views

- Generell förklaring av vyer:
 - <https://www.youtube.com/watch?v=OP6zvaRdkuw>

Övningar - Views

- Övningarna finns i repot tillsammans med lösningsförslag.
- Jobba tillsammans i grupper och gärna mellan grupperna om det behövs.

Förberedelser inför nästa tillfälle

- Aggregerade funktioner
<https://www.youtube.com/watch?v=0pfKXxB6aD8>
- Hjälp om SQL:
<http://www.mysqltutorial.org/>