Databasteknik

Aggregerade, Subqueries, SP

Utbildare: Mikael Olsson

mikael.olsson@emmio.se
076-174 90 43

NACKADEMIN

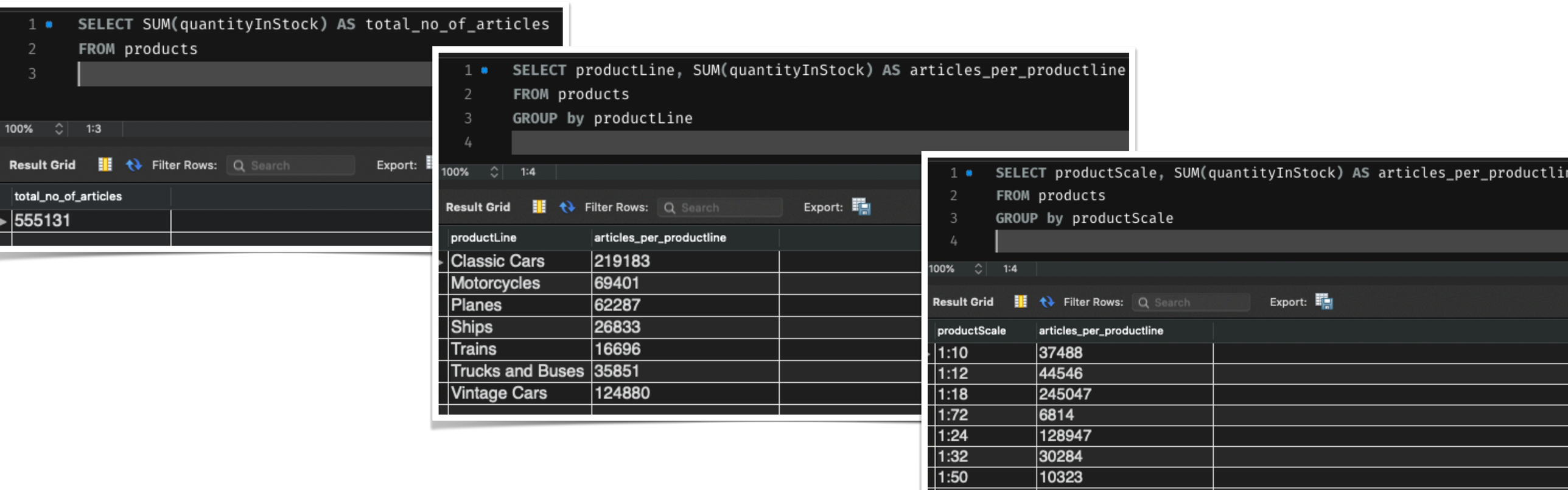
Socrative

https://www.socrative.com/

- Frågehanterare
  - Logga in som student
  - Ange rum “Emmio”
  - Få upp en vänta-skärm
  - Quiz: Join & Views

Aggregerade funktioner
Group by

- Alla aggregerad funktioner kan användas som de är eller beräknade per delmängd.



Aggregerade funktioner

- `AVG` - Return the average of non-NULL values.
- `COUNT` - Return the number of rows in a group, including rows with NULL values.
- `GROUP_CONCAT` - Return a concatenated string.
- `MAX` - Return the highest value (maximum) in a set of non-NULL values.
- `MIN` - Return the lowest value (minimum) in a set of non-NULL values.
- `SUM` - Return the summation of all non-NULL values a set.

- Vi ska kolla på några exempel från https://www.mysqltutorial.org/mysql-aggregate-functions.aspx men jag visar gärna fler exempel om något område känns extra knepigt.

Jämförelse-funktioner

- `COALESCE` – return the first non-NULL arguments, which is very handy for substitution ofNULL.
- `GREATEST & LEAST` – take n arguments and return the greatest and least values of the narguments respectively.
- `ISNULL` – return 1 if the argument is NULL, otherwise, return zero.
- https://www.mysqltutorial.org/mysql-comparison-functions/

Flödeskontroll

- `CASE` - return the corresponding result in `THEN` branch if the condition in the `WHEN` branch is satisfied, otherwise, return the result in the `ELSE` branch.
- `IF` - return a value based on a given condition.
- `IFNULL` - return the first argument if it is not NULL , otherwise returns the second argument.
- `NULLIF` - return NULL if the first argument is equal to the second argument, otherwise, returns the first argument.
- https://www.mysqltutorial.org/mysql-control-flow-functions/

Datumfunktioner

- `CURDATE` - Returns the current date.
- `DATEDIFF` - Calculates the number of days between two DATE values.
- `DAY` - Gets the day of the month of a specified date.
- `DATE_ADD` - Adds a time value to date value.
- `DATE_SUB` - Subtracts a time value from a date value.
- `DATE_FORMAT` - Formats a date value based on a specified date format.
- `DAYNAME` - Gets the name of a weekday for a specified date.
- `DAYOFWEEK` - Returns the weekday index for a date.
- `EXTRACT` - Extracts a part of a date.
- `LAST_DAY` - Returns the last day of the month of a specified date
- `NOW` - Returns the current date and time at which the statement executed.
- `MONTH` - Returns an integer that represents a month of a specified date.
- `STR_TO_DATE` - Converts a string into a date and time value based on a specified format.
- `SYSDATE` - Returns the current date.
- `TIMEDIFF` - Calculates the difference between two TIME or DATETIME values.
- `TIMESTAMPDIFF` - Calculates the difference between two DATE or DATETIME values.
- `WEEK` - Returns a week number of a date.
- `WEEKDAY` - Returns a weekday index for a date.
- `YEAR` - Return the year for a specified date

https://www.mysqltutorial.org/mysql-date-functions/

Sträng-funktioner

- `CONCAT` - Concatenate two or more strings into a single string
- `INSTR` - Return the position of the first occurrence of a substring in a string
- `LENGTH` - Get the length of a string in bytes and in characters
- `LEFT` - Get a specified number of leftmost characters from a string
- `LOWER` - Convert a string to lowercase
- `LTRIM` - Remove all leading spaces from a string
- `REPLACE` - Search and replace a substring in a string
- `RIGHT` - Get a specified number of rightmost characters from a string
- `RTRIM` - Remove all trailing spaces from a string
- `SUBSTRING` - Extract a substring starting from a position with a specific length.
- `SUBSTRING_INDEX` - Return a substring from a string before a specified number of occurrences of a delimiter
- `TRIM` - Remove unwanted characters from a string.
- `FIND_IN_SET` - Find a string within a comma-separated list of strings
- `FORMAT` - Format a number with a specific locale, rounded to the number of decimals
- `UPPER` - Convert a string to uppercase
- https://www.mysqltutorial.org/mysql-string-functions/

Matematik-funktioner

- `ABS` - Returns the absolute value of a number
- `CEIL` - Returns the smallest integer value greater than or equal to the input number (n).
- `FLOOR` - Returns the largest integer value not greater than the argument
- `MOD` - Returns the remainder of a number divided by another
- `ROUND` - Rounds a number to a specified number of decimal places.
- `TRUNCATE` - Truncates a number to a specified number of decimal places
- https://www.mysqltutorial.org/mysql-math-functions/

Subqueries

- *Predicate subqueries* - utökade logiska konstruktioner i `WHERE`- och `HAVING`-delarna.
- *Skalära subqueries* - fristående frågor som returnerar ett enda värde.

Predicate subqueries

- Kan enbart användas i `WHERE`- och `HAVING`-delarna. Måste returnera en kolumn.

```
mysql> SELECT * FROM Client WHERE C_ID IN
    -> (SELECT C_ID FROM Products);
+------+--------------+----------+
| C_ID | Name         | City     |
+------+--------------+----------+
| 1    | A K Ltd      | Delhi    |
| 2    | V K Associate | Mumbai   |
| 3    | R K India    | Banglore |
| 5    | A T Ltd      | Delhi    |
+------+--------------+----------+
4 rows in set (0.00 sec)
```

Skalära subqueries

- Returnerar ett enda värde.
- Kan användas där en kolumn kan användas.

```sql
SELECT
  c.customerName,
  (SELECT SUM(amount)
   FROM payments
   WHERE customerNumber = c.customerNumber) AS TotalPaid
FROM customers c;
```

verview | Output | Snippets | Result (1) ×

Fetched 122 records. Duration: 0.

customerName	TotalPaid
Atelier graphique	22314.36
Signal Gift Stores	80180.98
Australian Collectors, Co.	180585.06999999998
La Rochelle Gifts	116949.68000000001
Baane Mini Imports	104224.79

HAVING

- Hur visar vi enbart rader där avgQuantityInStock > 3500?

```
1 •  SELECT productLine, AVG(quantityInStock) AS avgQuantityInStock
2    FROM products
3    GROUP BY productLine;
```

Result (1) Result (1) Result (1) Result (1) Result (1) Result (1) Result (1) Res

Fetched 7 records. Duration: 0.000

productLine	avgQuantityInStock
Classic Cars	5767.9737
Motorcycles	5338.5385
Planes	5190.5833
Ships	2981.4444
Trains	5565.3333
Trucks and Buses	3259.1818
Vintage Cars	5203.3333

HAVING

WHERE funkar inte som vi kanske tror i aggregerade funktioner

```
1  SELECT productLine, AVG(quantityInStock) AS avgQuantityInStock
2  FROM products
3  WHERE avgQuantityInStock > 3500
4  GROUP BY productLine;
```

⊗	249	22:12:10	Error Code: 1054Unknown column 'avgQuantityInStock' in 'where clause'

```
1  SELECT productLine, AVG(quantityInStock) AS avgQuantityInStock
2  FROM products
3  WHERE AVG(quantityInStock) > 3500
4  GROUP BY productLine;
```

⊗	250	22:14:26	Error Code: 1111Invalid use of group function

HAVING

```sql
SELECT productLine, AVG(quantityInStock) AS avgQuantityInStock
FROM products
GROUP BY productLine
HAVING avgQuantityInStock > 3500;
```

esult (1) Result (1) Result (1) Result (1) Result (1) Result (1) Result (1) Re

Fetched 5 records. Duration: 0.00

productLine	avgQuantityInStock
Classic Cars	5767.9737
Motorcycles	5338.5385
Planes	5190.5833
Trains	5565.3333
Vintage Cars	5203.3333

HAVING

- `WHERE` används för att begränsa rader.
  - Används även för att avgöra vilka tabeller och index som ska användas.
- `HAVING` är ett “filter” på resultatet
  - Läggs på efter `ORDER BY` och `GROUP BY`.
- `WHERE` ger bättre performance än `HAVING`.

Stored Procedures

- Batches
- Stored procedures
- Lokala och globala variabler
- Parametrar

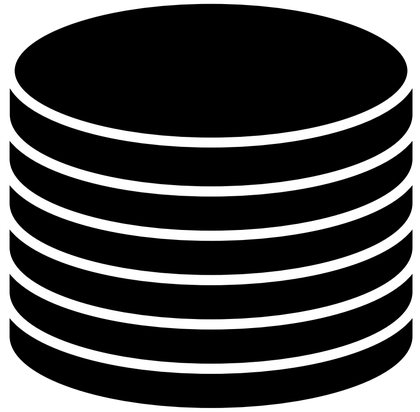
Stored Procedures

- Procedurkod kan lagras i ”stored procedures” på servern.
- Procedurkod kan också skickas från klienten till servern för att köras där.

Stored Procedures

Batch

- En batch är två eller flera SQL-satser
  - Skickas ihop, som ett nätverkspaket.
    - Reducerar antalet anrop
  - Analyseras tillsammans.



```
SELECT * FROM products;
SELECT * FROM orders;
```

USE

- USE *databasnamn*
  - USE ändrar aktuell databas
    - Som att dubbelklicka i WorkBench

```
USE classicmodels;
SELECT * FROM products;

USE komplit_ikt;
SELECT * FROM amne;
```

Regler för batchar

- Vissa satser måste ligga i en egen batch.
  - T ex `CREATE PROCEDURE`
- För att köra en lagrad procedur använder man `CALL`.
  - I MSSQL: `EXEC`
- Man kan inte ta bort och återskapa tabell i samma batch.
- Lokala variabler gäller bara i en batch.

Skript

- Ett skript är en batch som lagras i en fil.
  - Kan köras från Workbench eller kommandoraden.
- Det är vanligt att använda ett skript för att skapa en databas, dess tabeller och constraints.
- Exempel är *`classicmodels`*.

Stored procedure

- En lagrad procedur är förkompilerad och optimerad sql som sparas på servern.

Stored procedure

Fördelar

- Snabbare exekvering
  - Förkompilerade, optimerade, cachade i servern.
- Reducerad nätverkstrafik
  - SQL skickas inte längre över nätet.
- Bättre generalitet
  - Parametrar ger mångsidig kod
- Bättre säkerhet
  - Man kan tillåta exekvering av SP utan att tillåta access till underliggande tabeller.

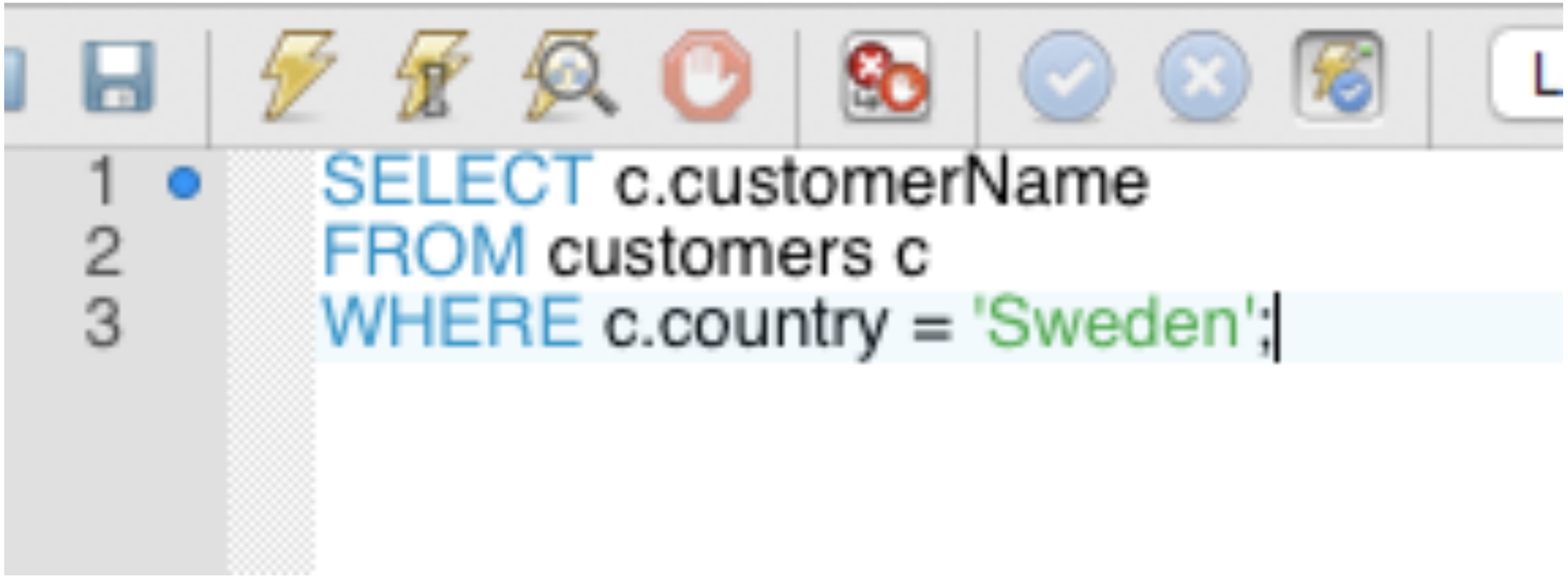
Stored procedure

Bättre prestanda

- En klient som kör flera SQL-frågor skapar extra nätverkstrafik och mer arbete för servern att tolka, validera och optimera varje sats.

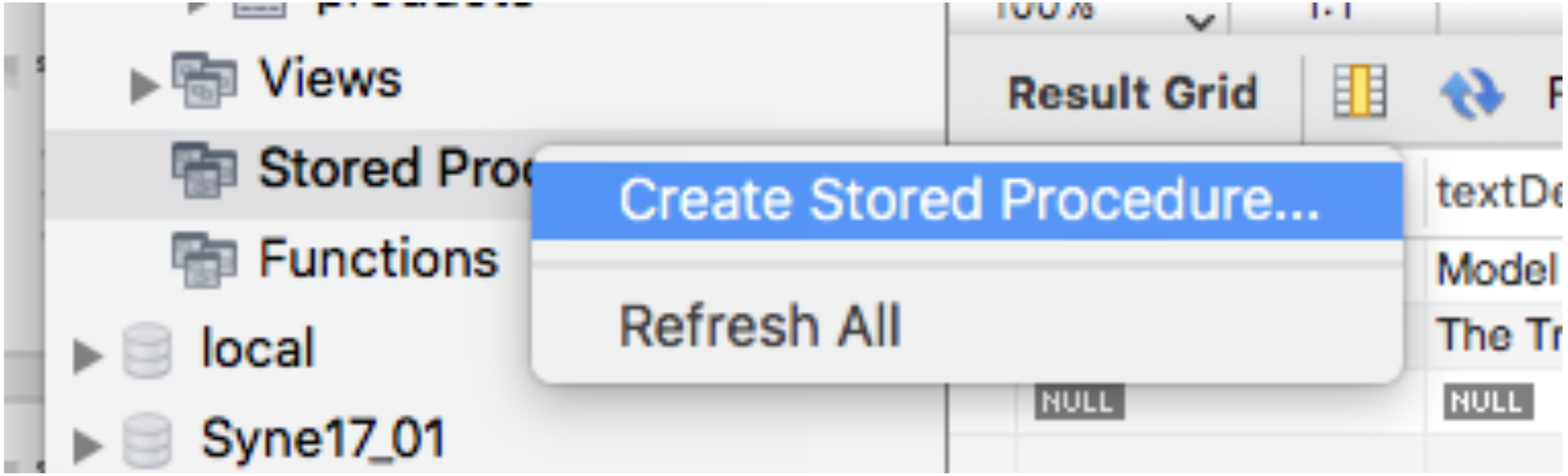
Stored procedure

Skapa och köra



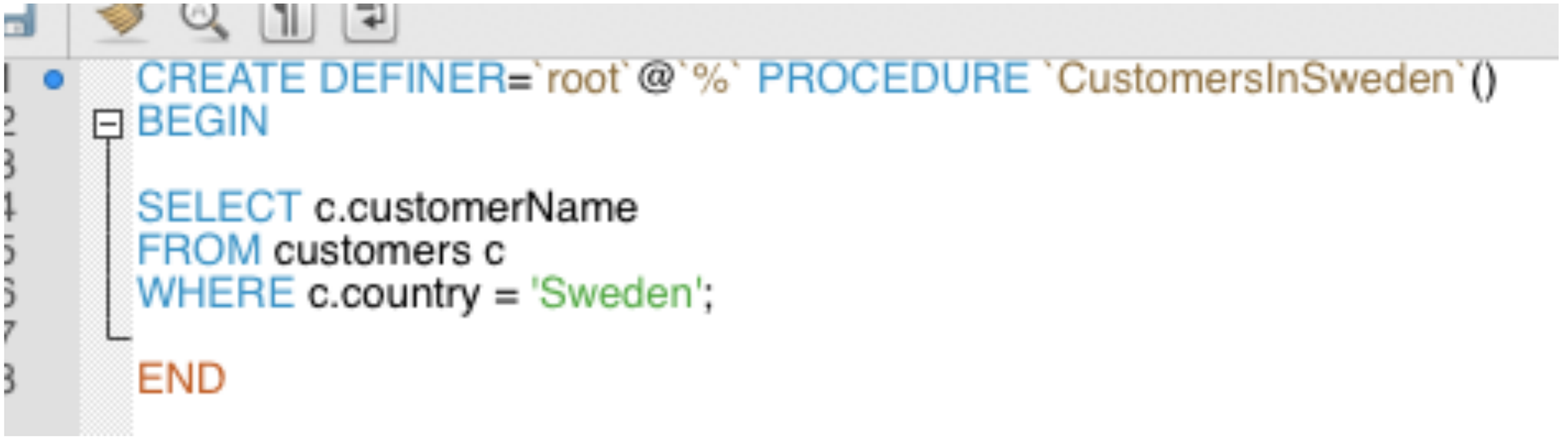
Stored procedure

Skapa och köra



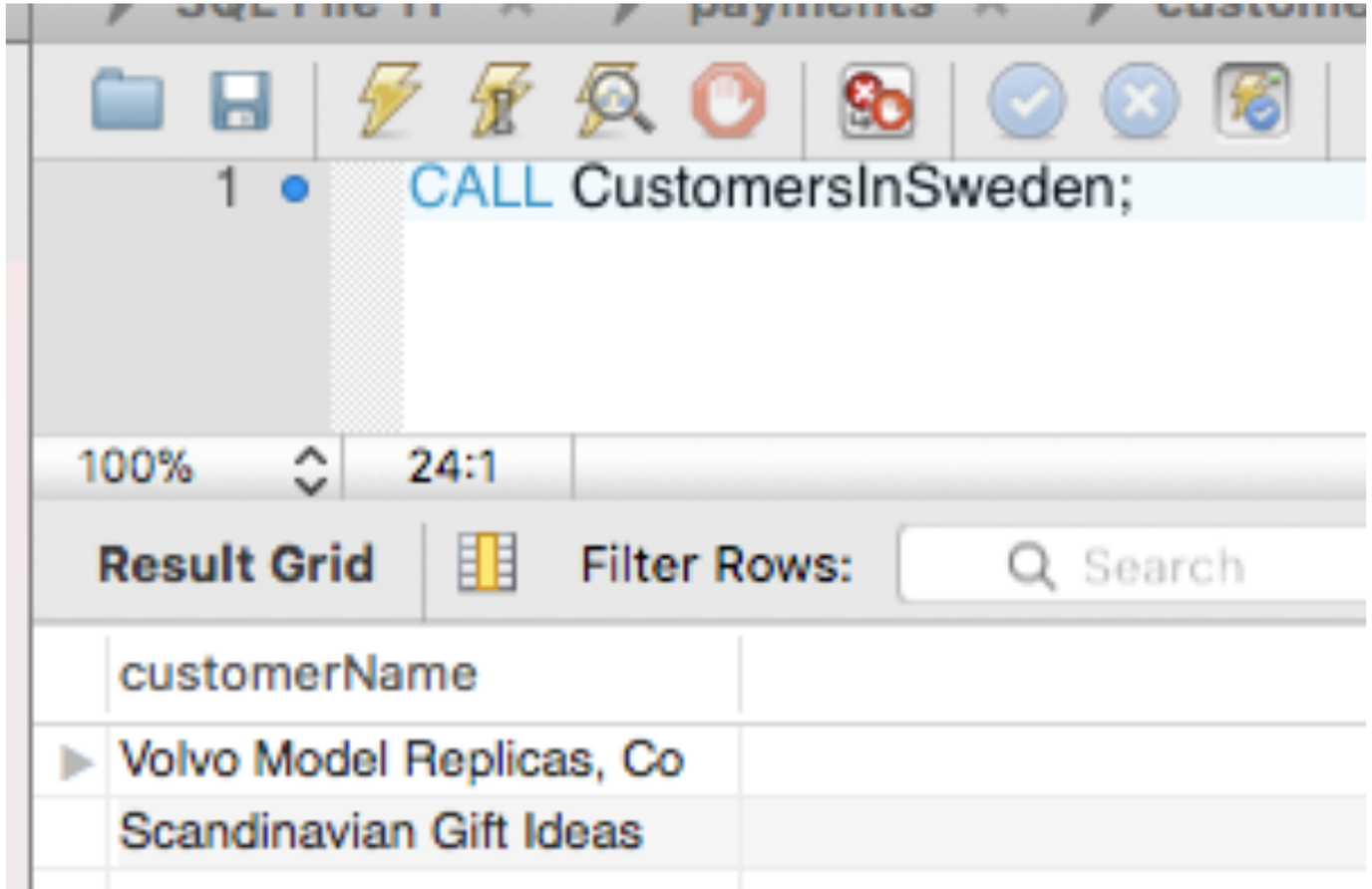
Stored procedure

Skapa och köra



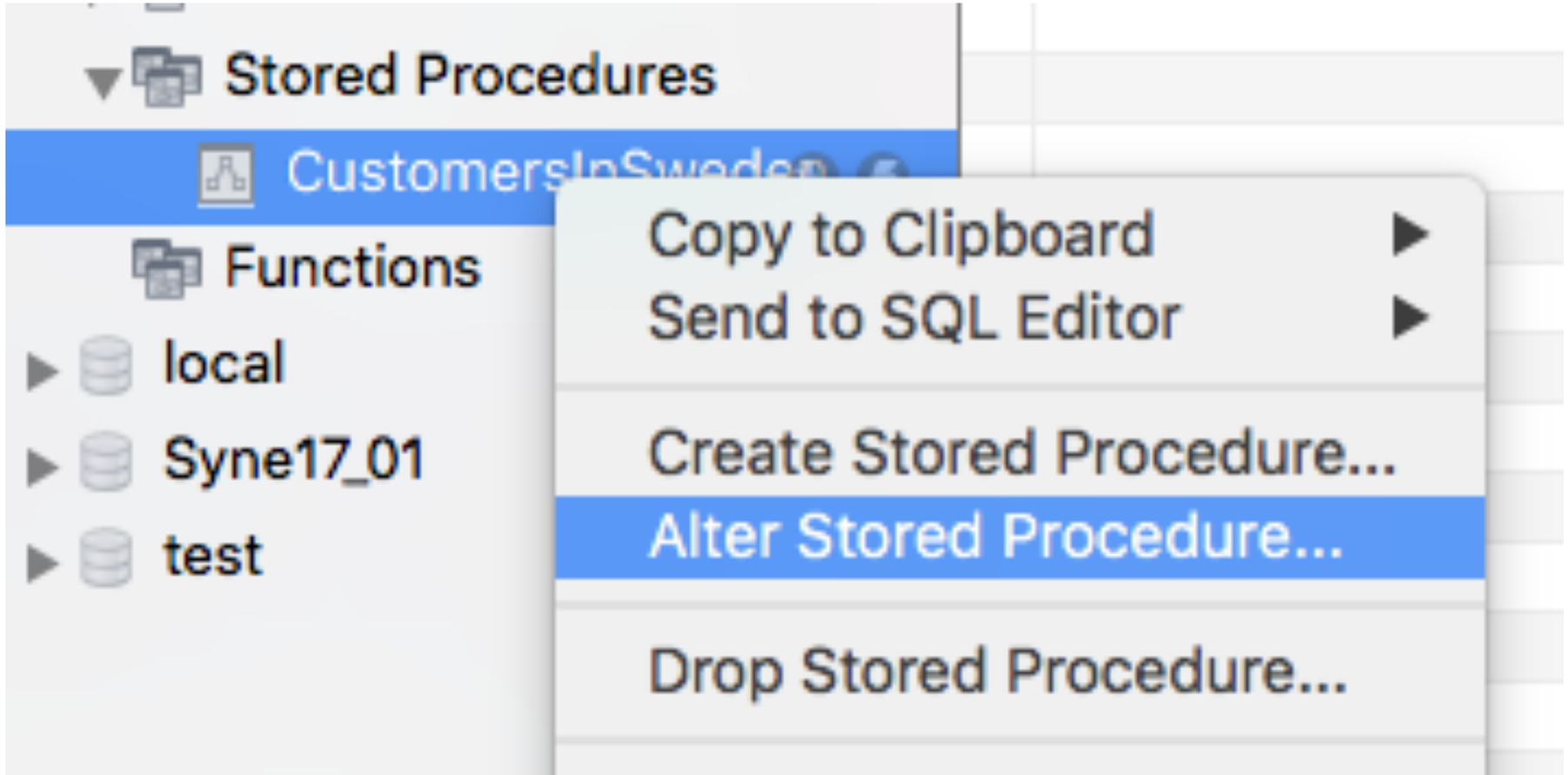
Stored procedure

Skapa och köra



Stored procedure

Ändra



Stored procedure

Lokala variabler

- För mer avancerade SP krävs variabler för att lagra delresultat.
  - `DECLARE variabel TYP;`
  - MSSQL: `DECLARE @variabel TYP;`

Stored procedure

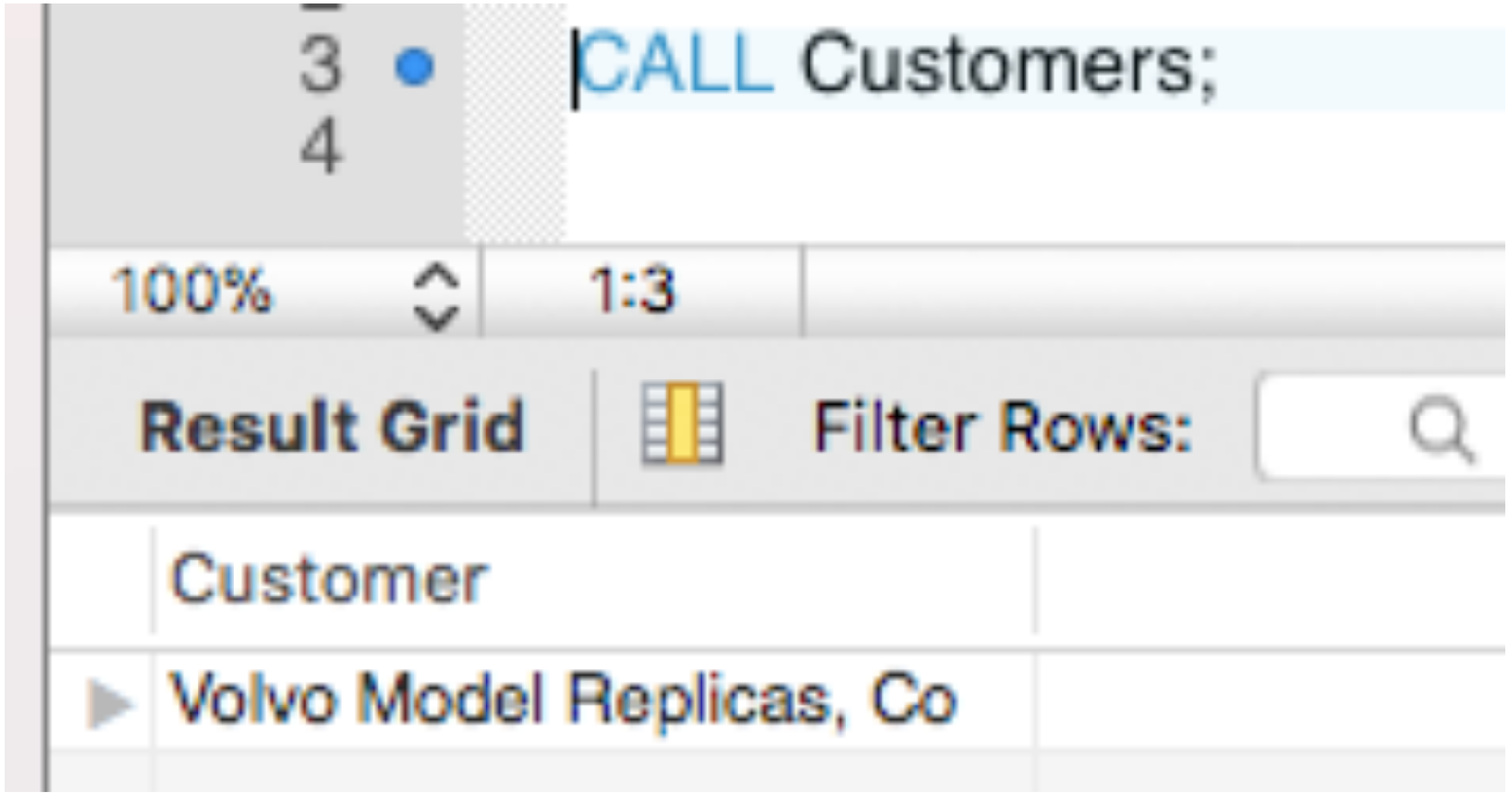
Lokala variabler

- För mer avancerade SP krävs variabler för att lagra delresultat.
  - `DECLARE variabel TYP;`

```
CREATE DEFINER=`root`@`%` PROCEDURE `Customers`()
BEGIN

DECLARE Customer VARCHAR(45);

SELECT c.customerName
FROM customers c
WHERE c.country = 'Sweden'
LIMIT 1
INTO Customer;

SELECT Customer;

END
```

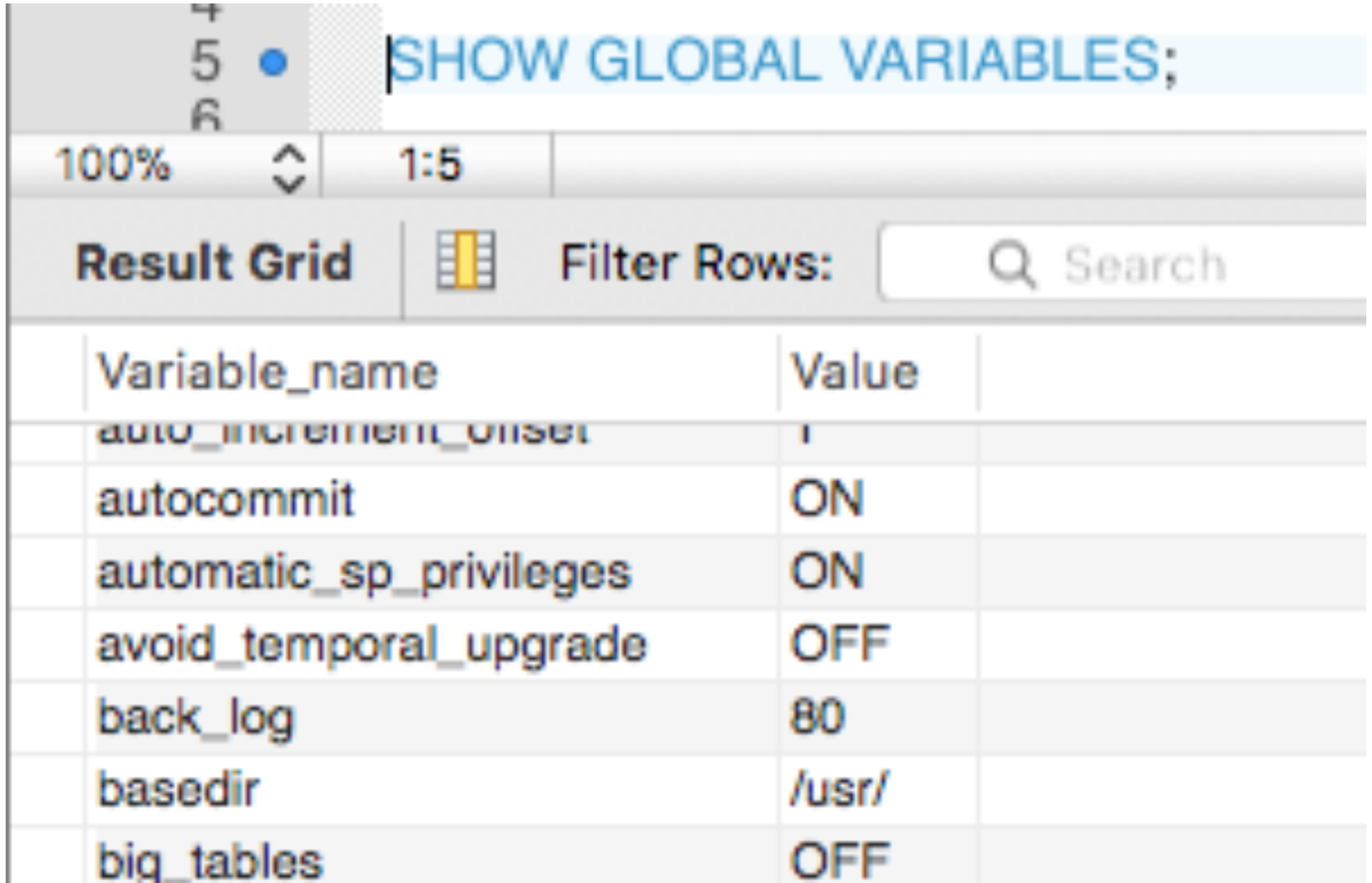
Stored procedure

Lokala variabler



Stored procedure

Tilldela variabler

- ```
  SELECT Customer = c.customerName
  FROM customers c
  WHERE country = 'Sweden';
  ```
- ```
  SET Customer = 'Volvo';
  ```
- ```
  SELECT Customer = 'Volvo';
  ```
  ```

Globala variabler



In-parametrar

```
CREATE DEFINER=`root`@`%` PROCEDURE `CustomersInCountry`(
  p_country VARCHAR(45)
)
BEGIN

SELECT c.customerName
FROM customers c
WHERE c.country = p_country;


END
```

```
CALL CustomersInCountry('Italy');
```

1:9

Filter Rows: Search

customerName
Amica Models & Co.
Rovelli Gifts
L'ordine Souveniers
Frau da Collezione

In-parametrar

Default-värden

- MSSQL har stöd för default-värden:
- ```
  CREATE PROCEDURE ProcName (
    @ParamName datatyp = default
  )
  ```
- MySQL har ännu inget stöd för default-värde.
  ```

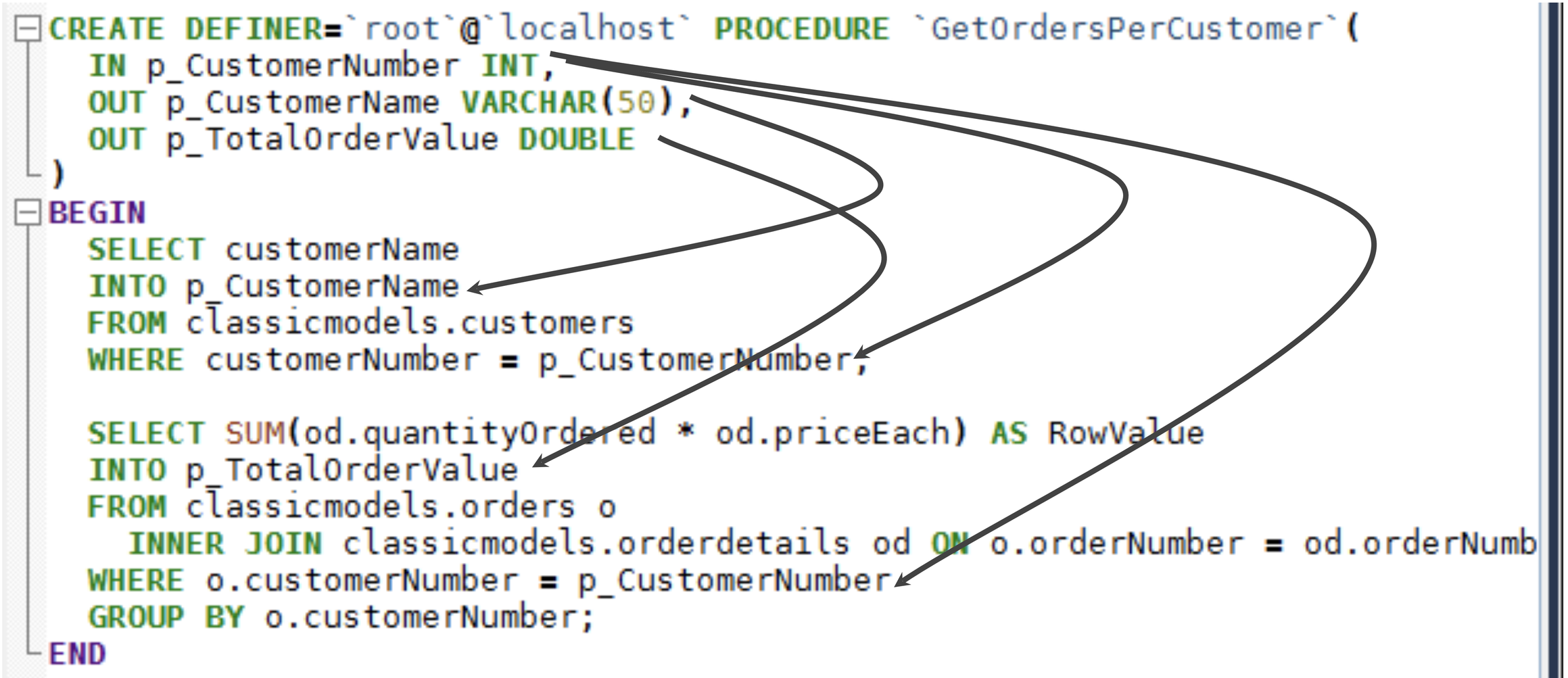
Parametrar

In & Out

- Parametrar kan vara av typ `IN`, `OUT` eller `INOUT`.
  - `IN` är ett värde du skickar som parameter till SP:n.
  - `OUT` är ett värde du får tillbaka från SP:n.
  - `INOUT` kan göra både och.

Parametrar

In & Out

- 

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `GetOrdersPerCustomer`(
    IN p_CustomerNumber INT,
    OUT p_CustomerName VARCHAR(50),
    OUT p_TotalOrderValue DOUBLE
)
BEGIN
    SELECT customerName
    INTO p_CustomerName
    FROM classicmodels.customers
    WHERE customerNumber = p_CustomerNumber;

    SELECT SUM(od.quantityOrdered * od.priceEach) AS RowValue
    INTO p_TotalOrderValue
    FROM classicmodels.orders o
        INNER JOIN classicmodels.orderdetails od ON o.orderNumber = od.orderNumb
    WHERE o.customerNumber = p_CustomerNumber
    GROUP BY o.customerNumber;
END
```

Hur man kör

- Variabel som innehåller returvärde.
- Måste starta med @. Förutom det gäller vanliga namngivningsregler.
- Använd `SELECT` för returvärden.

Hur man kör

```
 8
 9  CALL GetOrdersPerCustomer(128, @CustomerName, @TotalValue);
10
11  SELECT @CustomerName, @TotalValue;
```

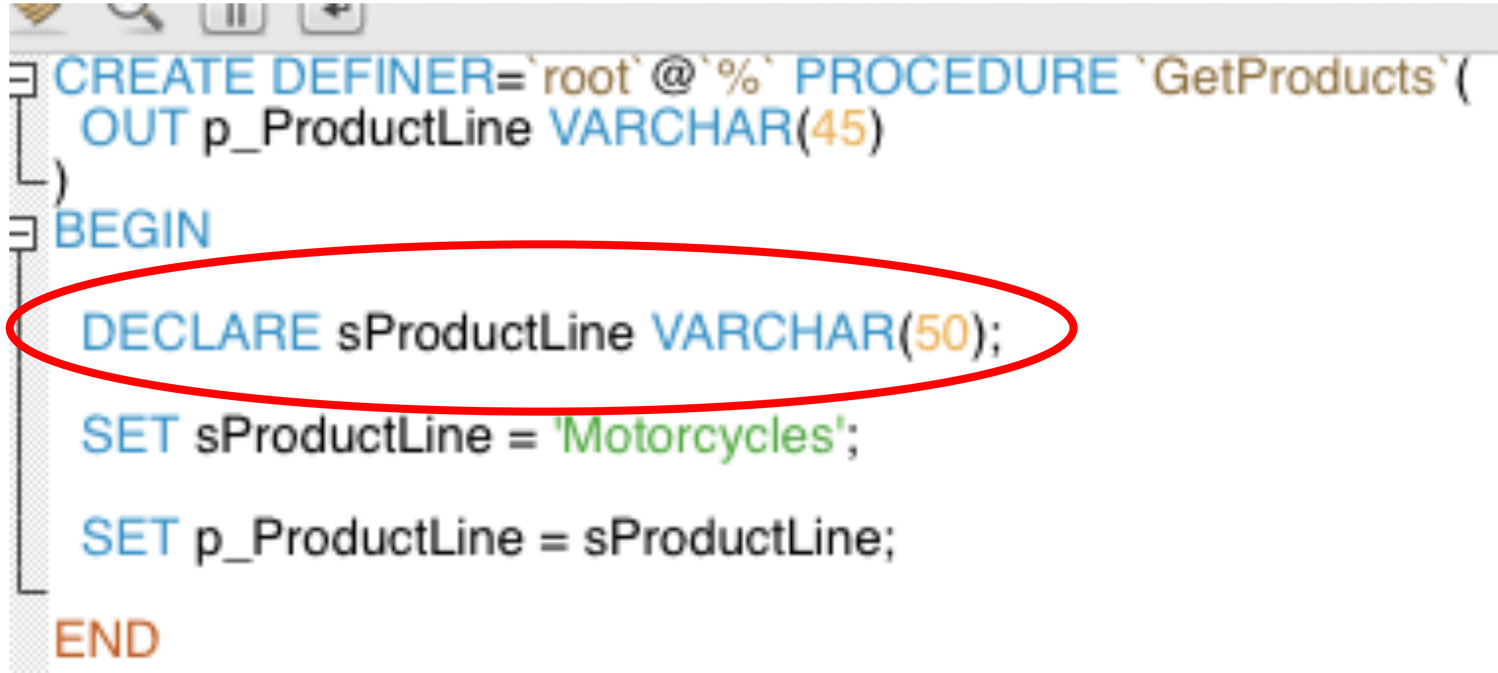
100% 35:11

Result Grid | Filter Rows: Search | Export:

@CustomerName	@TotalValue
Blauer See Auto, Co.	75937.76

DECLARE

- Du kan använda `DECLARE` för att deklarera en lokal variabel i en SP.
- Måste deklareras efter `BEGIN` men innan något annat statement.



Villkorade statements

```
IF expression THEN commands
    END IF;
```

```
IF expression THEN commands
    ELSE commands
    END IF;
```

```
IF expression THEN commands
    ELSEIF expression THEN commands
    ELSE commands
    END IF;
```

IF

```
CREATE DEFINER=`root`@`%` PROCEDURE `CheckStock`(
  IN p_ProductCode VARCHAR (15),
  OUT p_Status VARCHAR (45)
)
BEGIN

  DECLARE nQuantity INT;

  SELECT quantityInStock
  INTO nQuantity
  FROM products
  WHERE productCode = p_ProductCode;

  IF nQuantity < 100 THEN SET p_Status = "We're running low!";
    ELSE SET p_Status = "Plenty in stock!";
  END IF;

END
```

IF

```
13 •  CALL CheckStock ('S10_1678', @Message);
14 •  CALL CheckStock ('S12_1099', @Message2);
15
16 •  SELECT @Message, @Message2;
```

00% 1:16

Result Grid | Filter Rows: Search | E

@Message	@Message2
Plenty in stock!	We're running low!

Villkorade statements

```
1 CASE
2     WHEN expression THEN commands
3     …
4     WHEN expression THEN commands
5     ELSE commands
6     END CASE;
```

CASE statements

```
CREATE DEFINER=`root`@`%` PROCEDURE `CheckStock2`(
  IN p_ProductCode VARCHAR (15),
  OUT p_Status VARCHAR (45)
)
BEGIN

  DECLARE nQuantity INT;

  SELECT quantityInStock
  INTO nQuantity
  FROM products
  WHERE productCode = p_ProductCode;

  CASE
  WHEN nQuantity > 5000 THEN SET p_Status = "We have a lot of these!";
  WHEN nQuantity > 500 THEN SET p_Status = "We have a fair number of these!";
    ELSE SET p_Status = "We're running low on these.";
  END CASE;

END
```

CASE statements

productCode	quantityInStock
S10_1678	7933
S10_4757	3252
S12_1099	68

```
1  CALL CheckStock ('S10_1678', @Message,        @NoInStock1);
2  CALL CheckStock ('S12_1099', @AnotherMessage, @NoInStock2);
3  CALL CheckStock ('S10_4757', @ThirdMessage,   @NoInStock3);
4
5  SELECT @Message, @NoInStock1, @AnotherMessage, @NoInStock2, @ThirdMessage, @NoInStock3;
6
```

verview | Output | Snippets | Result (1) ×

Fetched 1 records. Duration: 0.000 sec, fetched in: 0.000 sec

@Message	@NoInStock1	@AnotherMessage	@NoInStock2	@ThirdMessage	@NoInStock3
We have a lot of these!	7933	We're starting to run low on these.	68	We have a fair number of these!	3252

LOOP - WHILE

```
01 DELIMITER $$
02 DROP PROCEDURE IF EXISTS WhileLoopProc$$
03 CREATE PROCEDURE WhileLoopProc()
04       BEGIN
05                 DECLARE x  INT;
06                 DECLARE str  VARCHAR(255);
07                 SET x = 1;
08                 SET str =  '';
09                 WHILE x  <= 5 DO
10                             SET  str = CONCAT(str,x,',');
11                             SET  x = x + 1;
12                 END WHILE;
13                 SELECT str;
14       END$$
15  DELIMITER ;
```

LOOP - REPEAT

```
01 DELIMITER $$
02 DROP PROCEDURE IF EXISTS RepeatLoopProc$$
03 CREATE PROCEDURE RepeatLoopProc()
04       BEGIN
05                DECLARE x  INT;
06                DECLARE str  VARCHAR(255);
07                SET x = 1;
08                SET str =  '';
09                REPEAT
10                          SET  str = CONCAT(str,x,',');
11                          SET  x = x + 1;
12                UNTIL x  > 5
13                END REPEAT;
14                SELECT str;
15       END$$
16 DELIMITER ;
```

LOOP - LEAVE & ITERATE

```
01 DELIMITER $$
02 DROP PROCEDURE IF EXISTS LOOPLoopProc$$
03 CREATE PROCEDURE LOOPLoopProc()
04       BEGIN
05               DECLARE x  INT;
06               DECLARE str  VARCHAR(255);
07               SET x = 1;
08               SET str =  '';
09               loop_label:  LOOP
10                           IF  x > 10 THEN
11                                       LEAVE  loop_label;
12                           END  IF;
13                           SET  x = x + 1;
14                           IF  (x mod 2) THEN
15                                       ITERATE  loop_label;
16                           ELSE
17                                       SET  str = CONCAT(str,x,',');
18                           END  IF;
19
20               END LOOP;
21               SELECT str;
22       END$$
23 DELIMITER ;
```

User Defined Functions

```
1  CREATE DEFINER=`root`@`%` FUNCTION `hello`(s CHAR(20)) RETURNS char(50) CHARSET utf8
2  BEGIN
3    RETURN CONCAT('Hello, ',s,'!');
4  END
```

```
18  SELECT hello(contactFirstName)
19  FROM customers;
20
```

100% 1:20

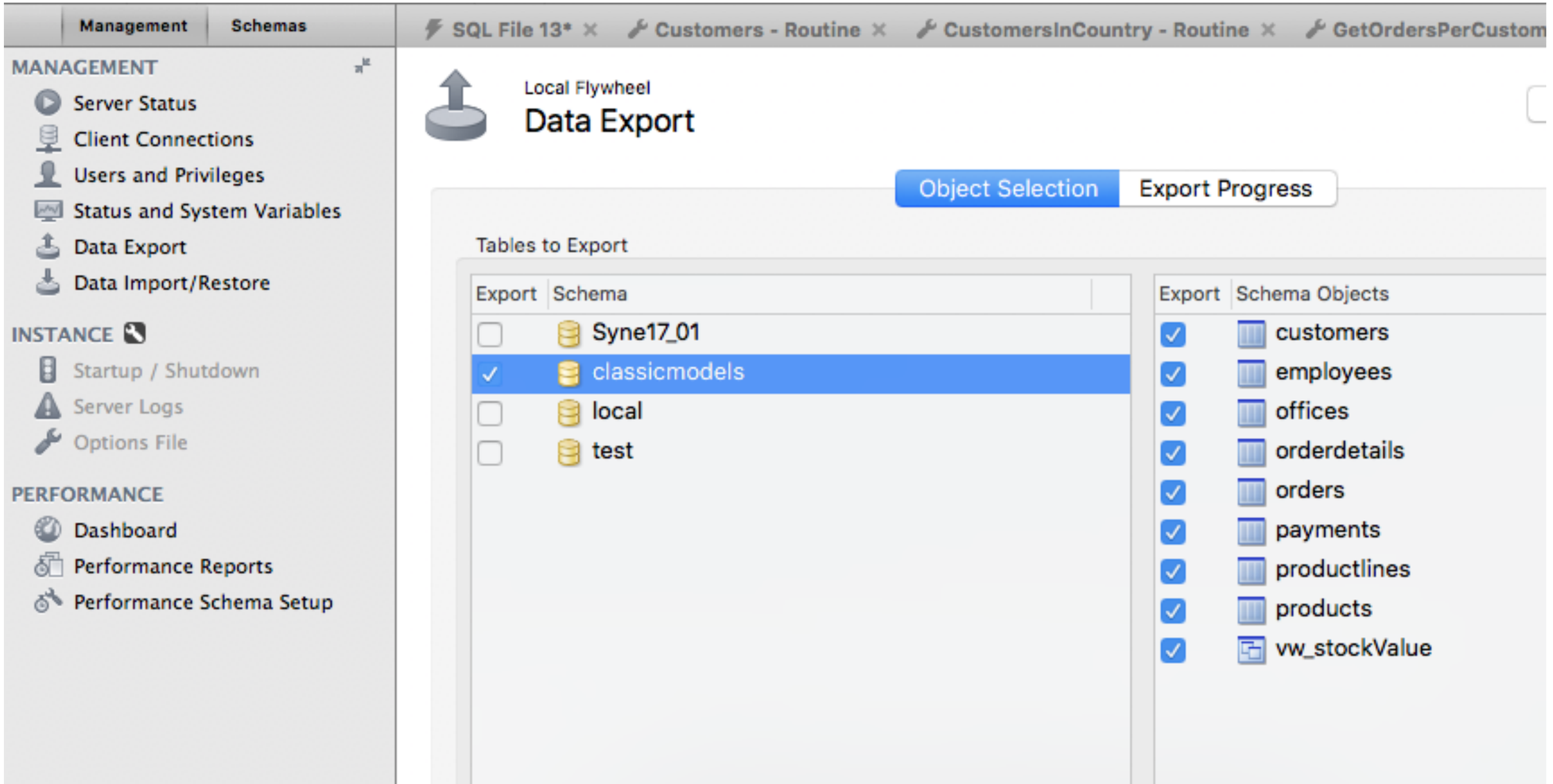
Result Grid | Filter Rows: | Search

hello(contactFirstName)
Hello, Jean!
Hello, Peter!
Hello, Janine!
Hello, Jonas!
Hello, Susan!

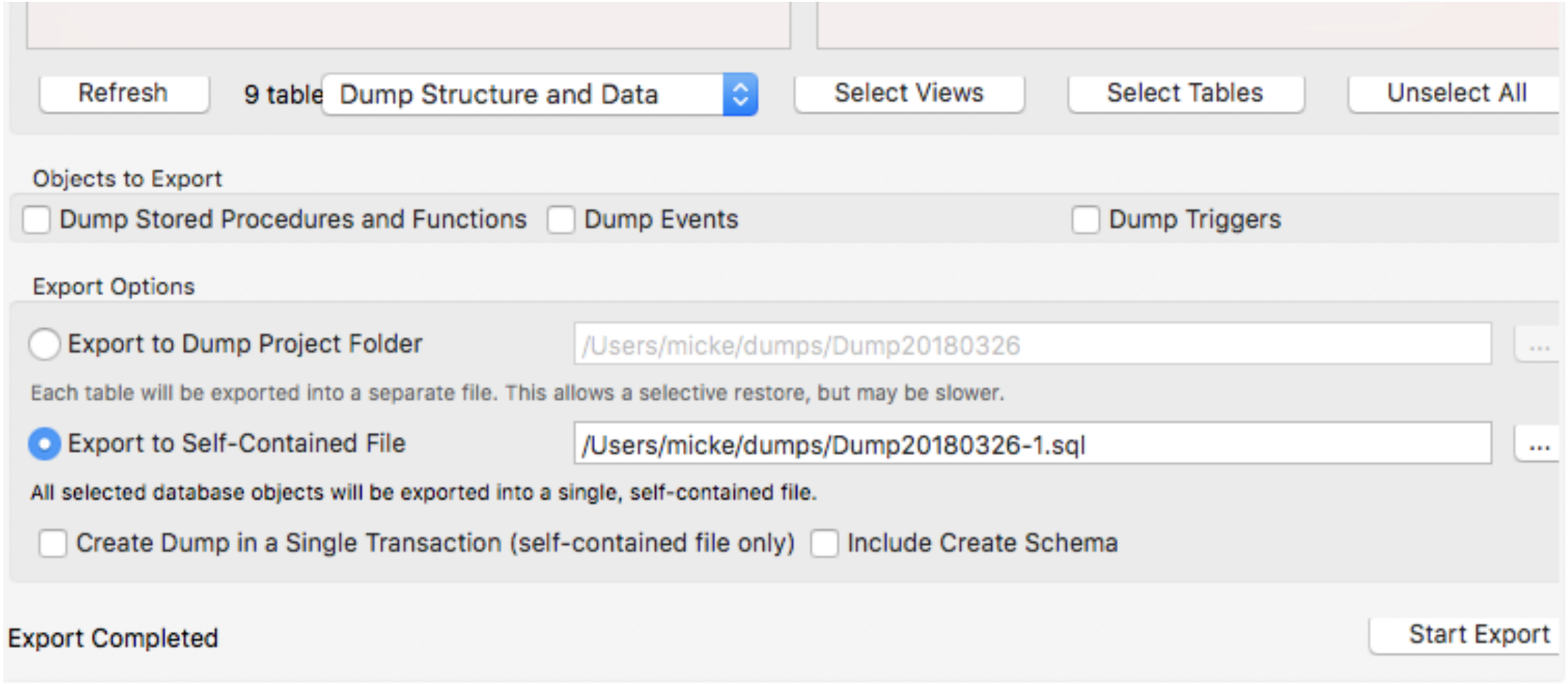
CLI vs GUI

- Command Line Interface
- Graphical User Interface
- När är vilket att föredra?

Exportera/importera data



Exportera/importera data



Exportera/importera data

- Self-contained ger en sql-fil likt `classic_models.sql`.

Övning 3

- Finns i repot.

Sammanfattning

- Aggregerade funktioner
  - Funktioner som gör beräkningar på grupper av data
- Funktioner
  - Några av de inbyggda funktionerna i MySQL
- Sub queries
  - Frågor med delfrågor eller underfrågor
- Skript
  - Automatiserade frågor
- Stored procedures
  - Kompilerade funktioner
- User Defined Functions
  - Egna funktioner
- Exportera / importera