PERTEMUAN 11
SQL (lanjutan): Queries, Constraints & Triggers

(Chap. 5 – Ramakrishnan)

Overview

Structure Query Language (SQL) adalah bahasa database relasional komersial yang paling banyak digunakan. SQL pada awalnya dikembangkan oleh IBM dalam SEQUEL-XRM dan Proyek System-R (1974-1977). Kemudian SQL berkembang mengikuti standar ANSI/ISO untuk SQL, yang disebut SQL-92.

Beberapa Aspek Bahasa SQL

- Data Definition Language (DDL) : subset SQL yang mendukung pembuatan, penghapusan, dan modifikasi struktur tabel beserta tampilannya.
- Data Manipulation Language (DML) : subset SQL dapat digunakan untuk menspesifikasikan queries, menyisipkan, menghapus, dan memodifikasi baris-baris tabel.
- Embedded dan Dinamic SQL : Fitur-fitur embedded SQL yang memungkinkan SQL untuk memanggil host language seperti C atau COBOL.
- Triggers : Standar SQL/1999 memberikan dukungan untuk triggers, yang bertindak secara otomatis dan memanipulasi database ketika kondisi terpenuhi.

Beberapa Aspek Bahasa SQL

- Security : SQL menyediakan mekanisme untuk mengendalikan akses pengguna ke objek database seperti tables dan views.

- Transaction Management : perintah SQL yang memungkinkan seseorang pengguna melakukan secara eksplisit untuk mengendalikan aspek, bagaimana sebuah transaksi harus dijalankan.

- Client-Server Execution & Remote Database Access : perintah-perintah SQL ini dapat digunakan untuk mengendalikan bagaimana suatu program aplikasi dapat dihubungkan ke sebuah SQL database server, atau mengakses data dari sebuah database melalui jaringan.

Contoh Instance

B1

bid	bname	color
101	Interlake	Blue
103	Clipper	Green

B2

bid	bname	color
102	Interlake	Red
104	Marine	Red

S1

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Contoh Instance

R1

Sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

S2

sid	sname	rating	age
28	Yuppy	9	35.0
31	Lubber	8	55.5
44	Guppy	5	35.0
58	Rusty	10	35.0

R2

Sid	bid	day
22	101	10/10/96
58	103	10/12/96

Bentuk Dasar SQL Query

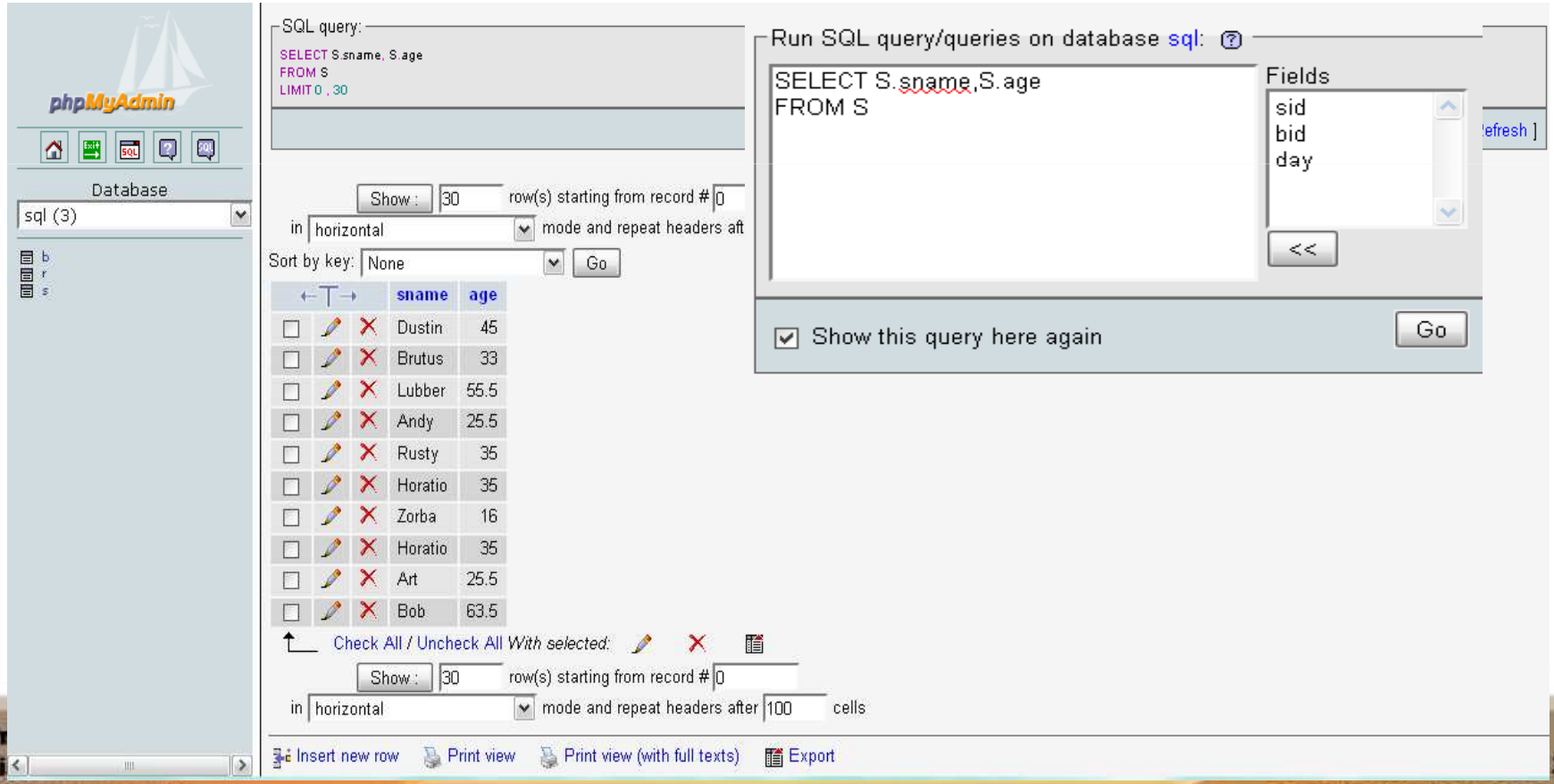
```
SELECT [DISTINCT] select-list
 FROM from-list
WHERE qualification
```

- *FROM-list* : Sebuah nama tabel dapat diikuti oleh berbagai variabel yang sangat berguna ketika nama tabel yang sama muncul lebih dari sekali dalam daftar.
- *Select-list* : list dari (ekspresi yang melibatkan) nama kolom atau field dari tabel.
- *Qualification* : klausa WHERE kombinasi boolean (AND, OR, dan NOT) di dalam bentuk ekspresi op ekspresi, dimana op adalah salah satu operator perbandingan {<, <=, =,< >,>=, >}.
- *DISTINCT* : *keyword* yang opsional. Hal ini menunjukkan bahwa tabel yang dihitung sebagai hasill dan tidak harus mengandung *duplicate,* yaitu dua baris data yang sama. Defaultnya adalah duplicate yang tidak dihilangkan.

Contoh Basic SQL Query

- Tampilkan nama dan umur dari tabel Sailors :

```
SELECT DISTINCT S.sname,S.age
FROM Sailors S
```



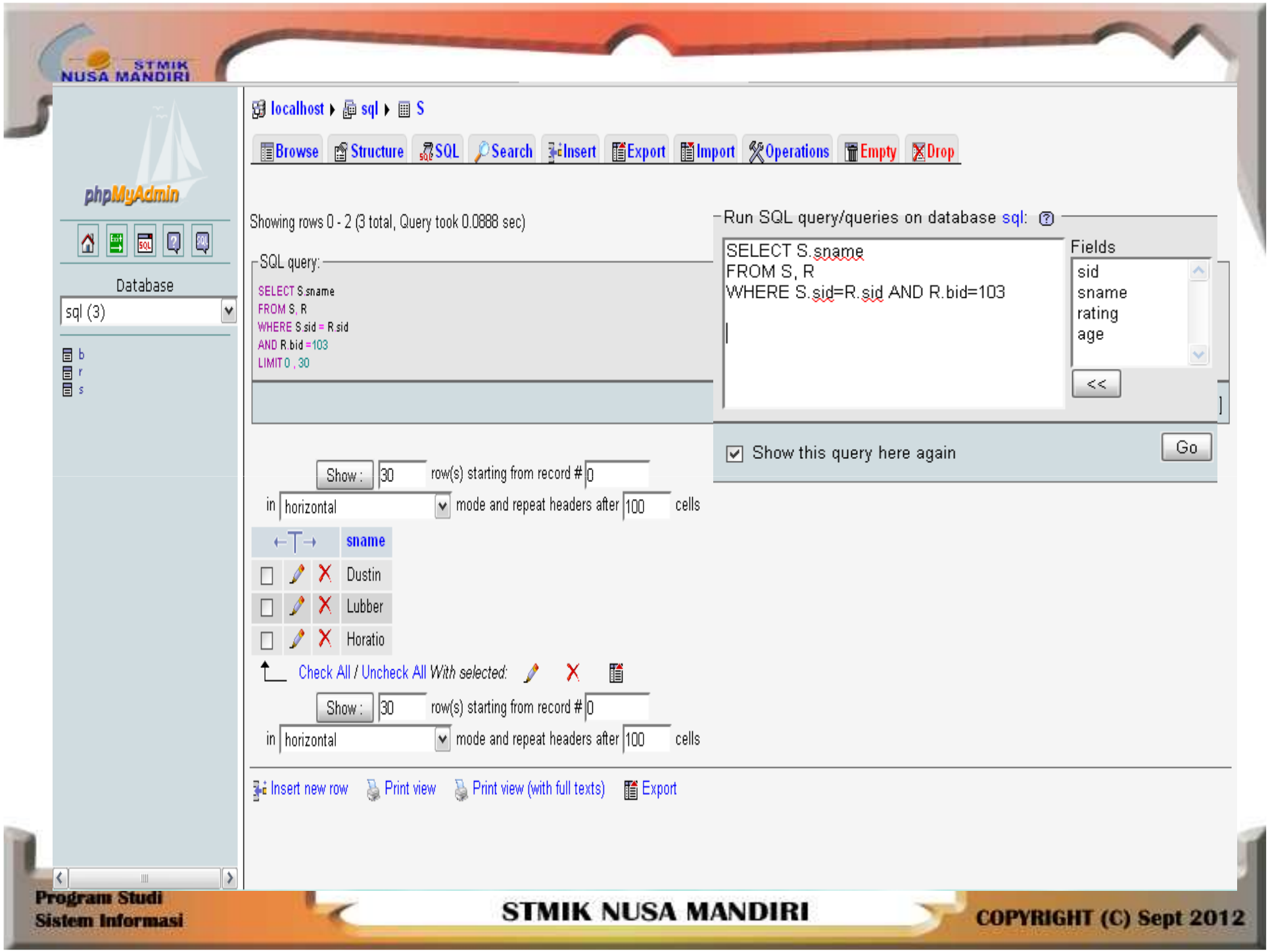
Strategi Evaluasi Konseptual

- Hitung hasil cross product dari form list.
- Hapus tuples hasil jika tuples tersebut tidak memenuhi qualifications.
- Hapus attributes yang tidak ada dalam select list.
- Jika digunakan DISTINCT, lakukan eliminasi baris-baris yang terduplikasi.

Contoh Strategi Konseptual :

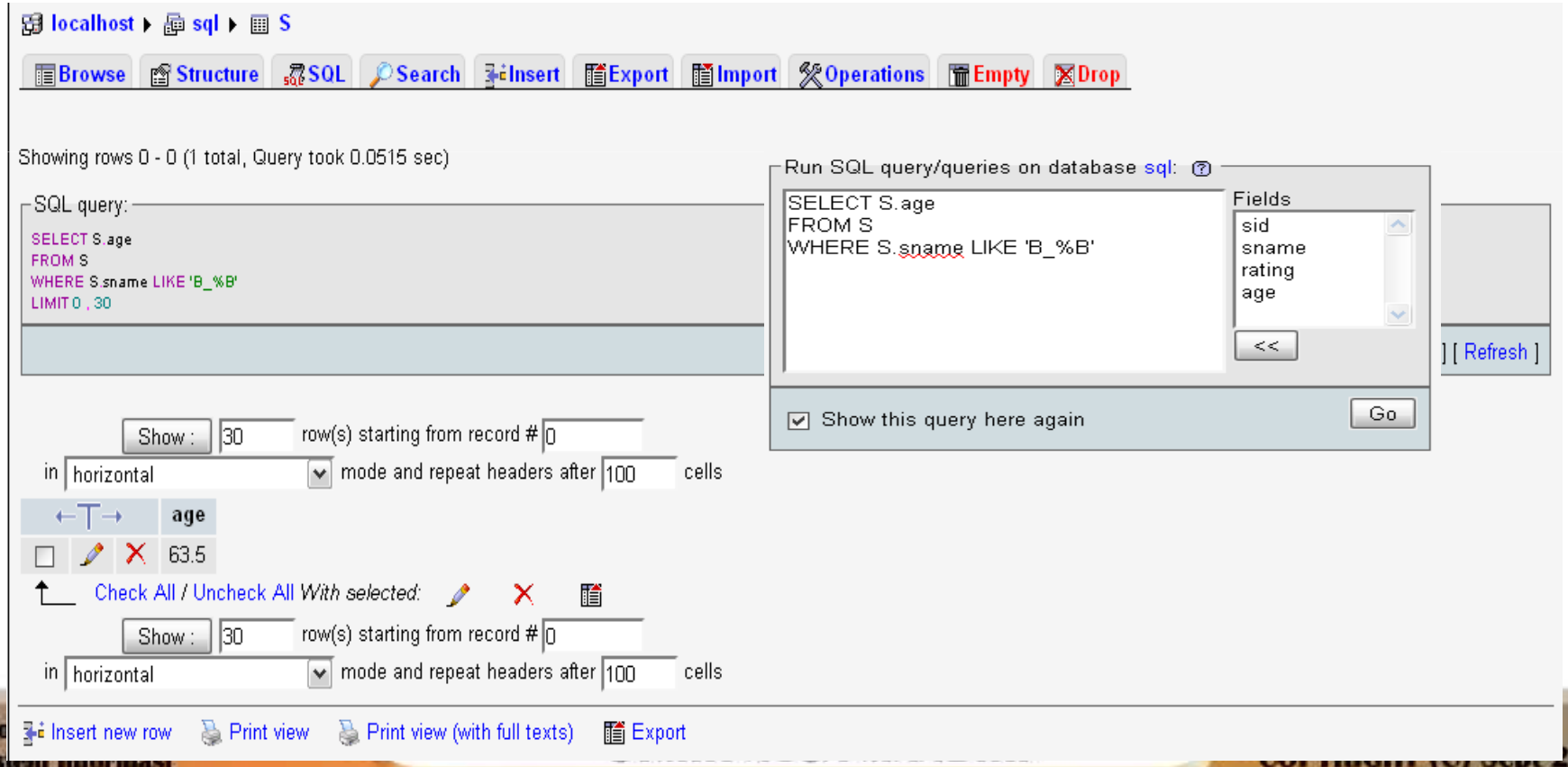
- Tentukan nama nama pelaut yang telah memesan sejumlah 103 perahu.

```
SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid=R.sid AND R.bid=103
```


STMIK NUSA MANDIRI
phpMyAdmin
Database
sql (3)
b
r
s
localhost ▸ sql ▸ S
Browse
Structure
SQL
Search
Insert
Export
Import
Operations
Empty
Drop
Showing rows 0 - 2 (3 total, Query took 0.0888 sec)
SQL query:
SELECT S.sname
FROM S, R
WHERE S.sid = R.sid
AND R.bid =103
LIMIT 0 , 30
Run SQL query/queries on database sql:
SELECT S.sname
FROM S, R
WHERE S.sid=R.sid AND R.bid=103
Fields
sid
sname
rating
age
<<
Show this query here again
Go
Show : 30 row(s) starting from record # 0
in horizontal mode and repeat headers after 100 cells
sname
Dustin
Lubber
Horatio
Check All / Uncheck All With selected:
Show : 30 row(s) starting from record # 0
in horizontal mode and repeat headers after 100 cells
Insert new row
Print view
Print view (with full texts)
Export
Program Studi Sistem Informasi
STMIK NUSA MANDIRI
COPYRIGHT (C) Sept 2012

Ekspresi & String

```
SELECT S.age
FROM Sailors S
WHERE S.sname LIKE 'B_%B'
```



Union, Intersect & Except

- UNION : dapat digunakan untuk menghitung union dari dua union-compatible yang merupakan kumpulan record dari hasil dua queries.
- INTERSECT : dapat digunakan untuk menghitung intersect dari dua intersect yang merupakan kumpulan dari record.
- Menampilkan name dari Sailors yang telah melakukan reservasi sebuah red boat atau green boat.

```
SELECT    S.sname
FROM      Sailors S, Reserves R, Boats B
WHERE     S.sid = R.sid AND R.bid = B.bid
AND (B.color = 'red' OR B.color = 'green')
```

```
SELECT    S.sname
FROM      Sailors S, Reserves R, Boats B
WHERE     S.sid = R.sid AND R.bid = B.bid AND
B.color = 'red'
UNION
SELECT    S2.sname
FROM      Sailors S2, Boats B2, Reserves R2
WHERE     S2.sid = R2.sid AND R2.bid = B2.bid AND
B2.color = 'green'
```

Union, Intersect & Except

- EXCEPT : dapat digunakan untuk menghitung *set difference* dari dua union-compatible yang merupakan kumpulan record dari hasil dua queries.

```
SELECT      S.sname
FROM        Sailors S, Reserves R, Boats B
WHERE       S.sid = R.sid AND R.bid = B.bid AND
B.color = ‘red’
INTERSECT
SELECT      S2.sname
FROM        Sailors S2, Boats B2, Reserves R2
WHERE       S2.sid = R2.sid AND R2.bid = B2.bid
AND B2.color = ‘green’
```

- Cari *sids* dari semua pelaut yang telah memesan Boat red tetapi tidak memesan Boat green.

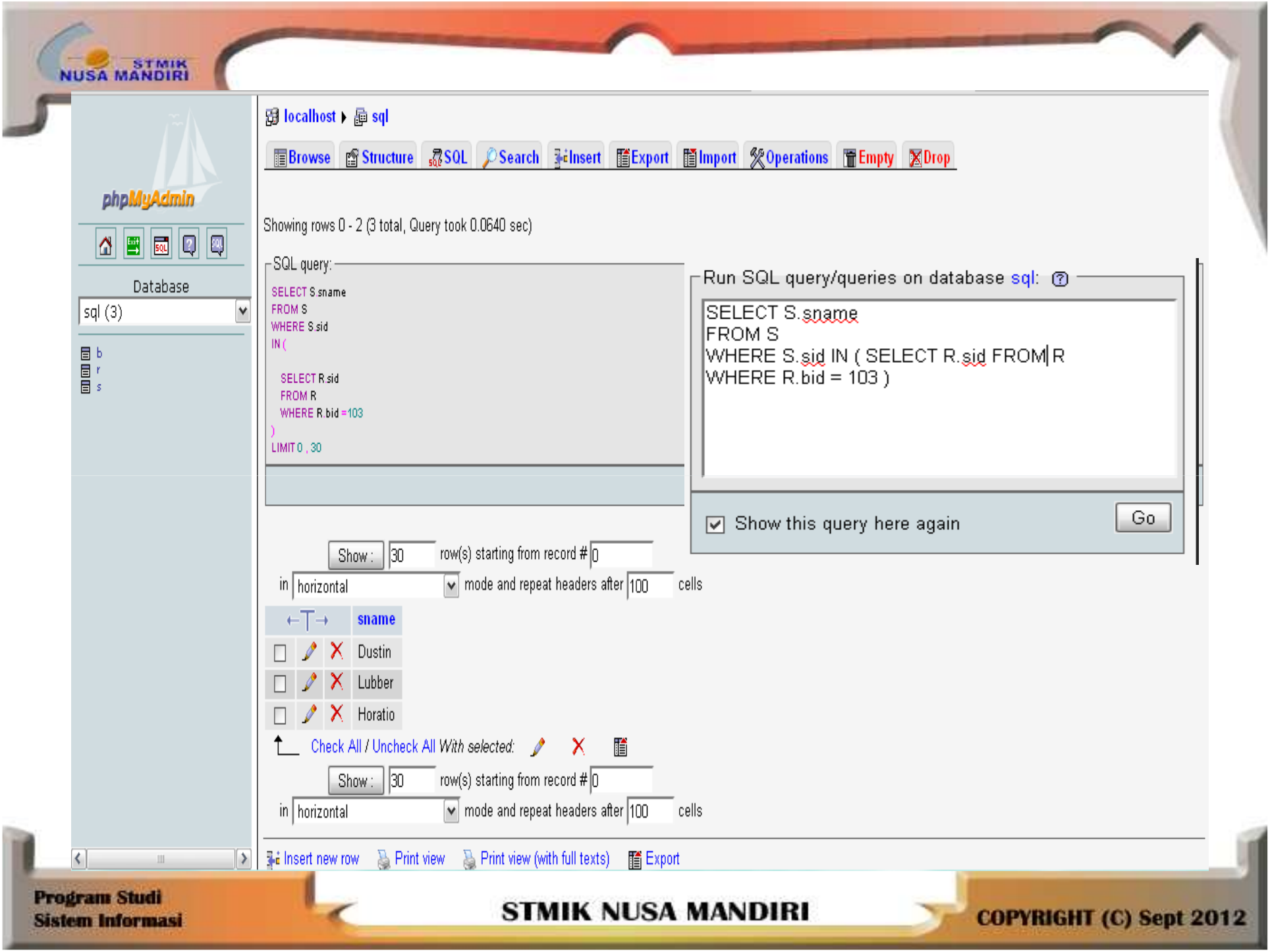
```
SELECT      S.sid
FROM        Sailors S, Reserves R, Boats B
WHERE       S.sid = R.sid AND R.bid = B.bid AND
B.color = ‘red’
EXCEPT
SELECT      S.sid
FROM  Sailors S2, Reserves R2, Boats B2
WHERE       S2.sid = R2.sid AND R2.bid = B2.bid
AND B2.color = ‘green’
```

Nested Queries

- Tentukan nama-nama pelaut yang telah memesan boat bernomor 103.

```
SELECT S.sname
FROM Sailors S
WHERE S.sid IN ( SELECT R.sid
                 FROM Reserves R
                 WHERE R.bid = 103 )
```

- Operator IN memungkinkan kita untuk menguji apakah nilai dalam himpunan elemen; sebuah query SQL yang digunakan untuk menghasilkan data pada query yang akan diuji.
- Untuk menampilkan name pada tabel Sailors yang tidak melakukan reservasi boat bernomor 103 gunakan NOT IN.

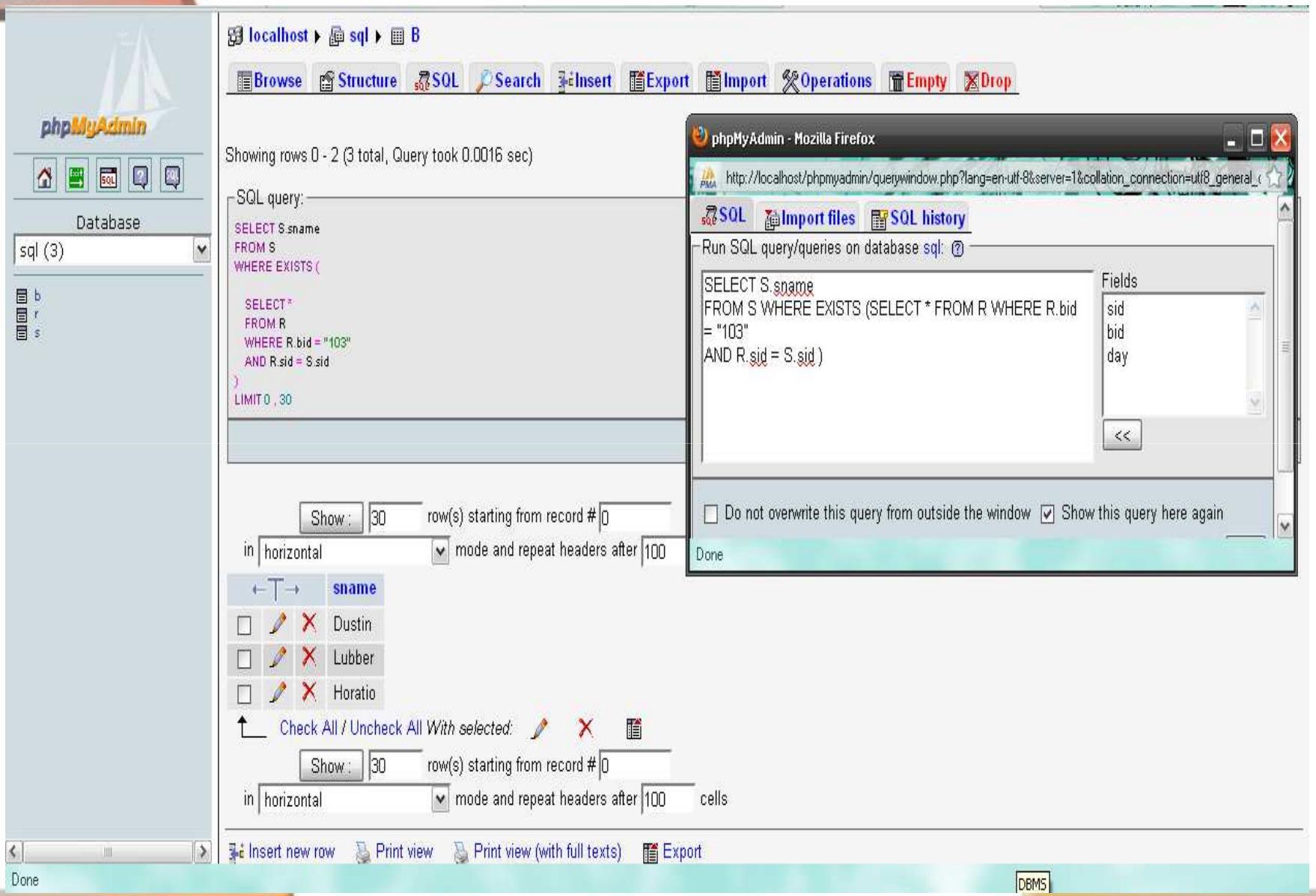


Korelasi Nested Queries

- Tentukan nama-nama sailors yang telah memesan boat bernomor 103

```
SELECT S.sname
FROM Sailors S
WHERE EXISTS ( SELECT * FROM Reserves R
                    WHERE R.bid = 103 AND R.sid = S.sid )
```

- EXIST : bentuk operator perbandingan yang jika bernlai TRUE akan dijadikan subquery sebagai parameter yang tidak menghilangkan set kosong.
- Untuk menampilkan name pada tabel Sailors yang tidak memesan boat bernomor 103 dengan menggunakan NOT EXIST.



Set-Comparison Operators

- Sebelumnya telah dibahas penggunaan EXIST, IN, dan UNIQUE. SQL juga mendukung op ANY dan op ALL, dimana op adalah salah satu operator perbandingan aritmatika {<, <=, =, <>,> =,>}. (SOME juga tersedia, tapi itu hanya sinonim untuk ANY)

- Contoh : Cari sailors yang mempunyai rating yang lebih besar dari sailors Horatio

```
SELECT  S.sid
FROM Sailors S
WHERE S.rating > ANY (SELECT S2.rating
        FROM Sailors S2
        WHERE S2.sname = 'Horatio')
```

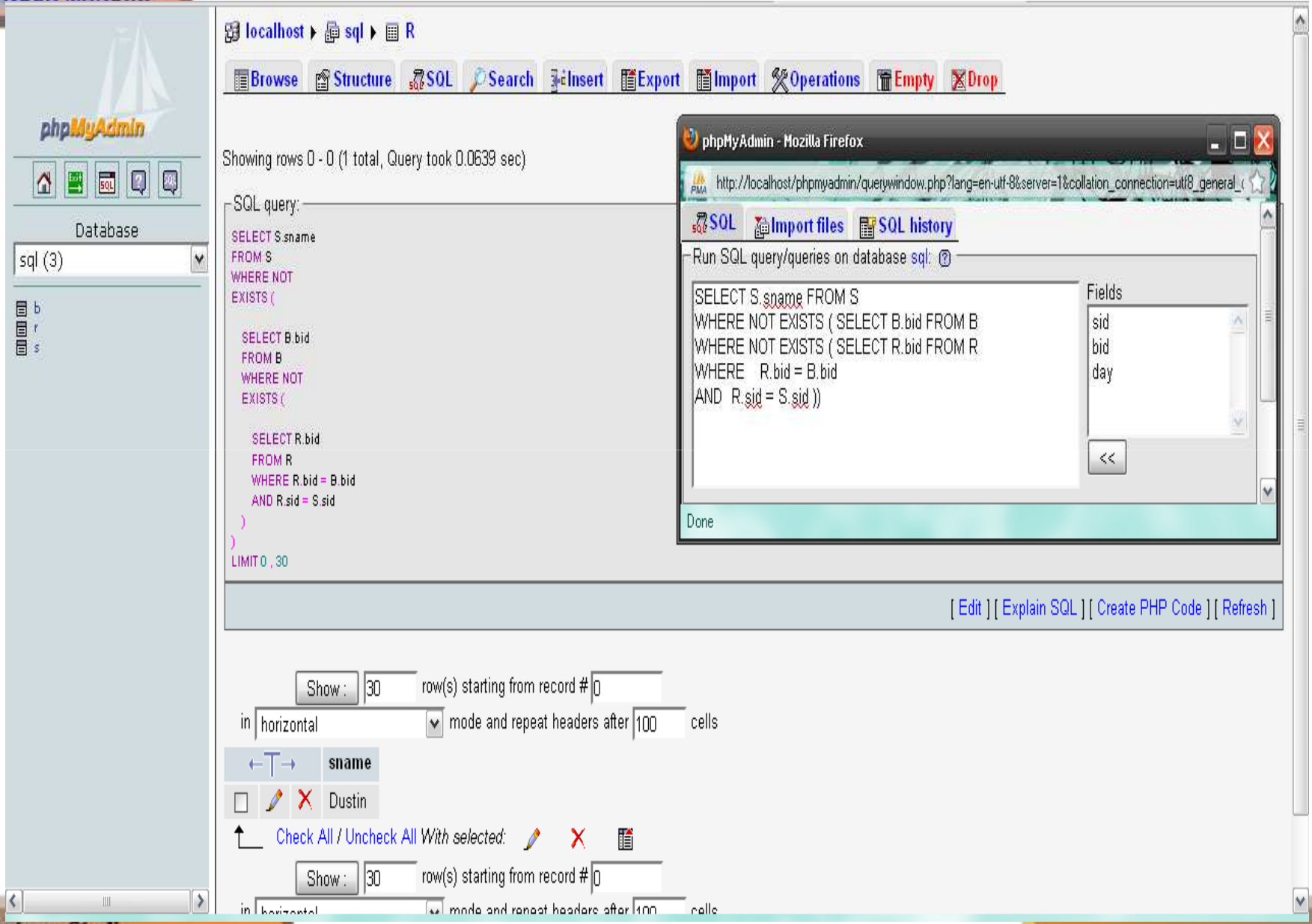
Division dalam SQL

- Tentukan nama sailors yang telah melakukan reservasi semua boats.
- Dengan menggunakan EXCEPT :

```
SELECT    S.sname
FROM      Sailors S
WHERE     NOT EXISTS (( SELECT B.bid FROM Boats B )
          EXCEPT
          (SELECT R.bid FROM Reserves R
                    WHERE R.sid = S.sid ))
```

- Cara yang lebih sulit tanpa menggunakan ECXEPT :

```
SELECT    S.sname
FROM      Sailors S
WHERE NOT EXISTS ( SELECT B.bid FROM Boats B
          WHERE NOT EXISTS ( SELECT R.bid
                    FROM Reserves R
                    WHERE R.bid = B.bid AND
                    R.sid = S.sid ))
```



Operator Agregasi

- COUNT ([DISTINCT] A) : Jumlah dari semua nilai-nilai dalam kolom A.
- SUM ([DISTINCT] A) : Jumlah dari semua (unik) nilai-nilai dalam kolom A.
- AVG ([DISTINCT] A) : Rata-rata dari semua (unik) nilai-nilai dalam kolom A.
- MAX (A) : Nilai maksimum di kolom A.
- MIN (A) : Nilai minimim dalam kolom A.

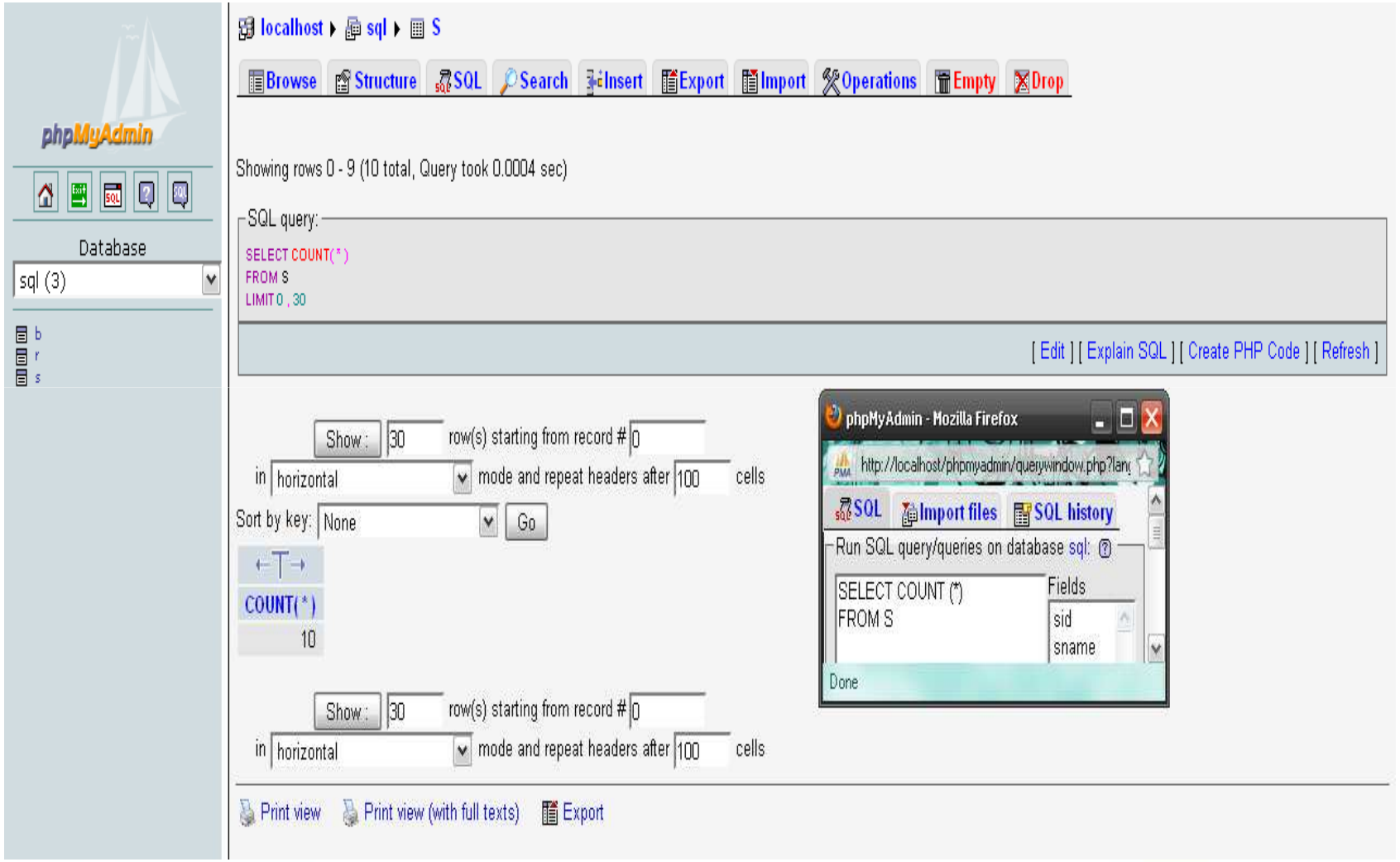
```
SELECT COUNT (*)
        FROM Sailors S
```

```
SELECT COUNT (DISTINCT S.sname)
                FROM Sailors S
```

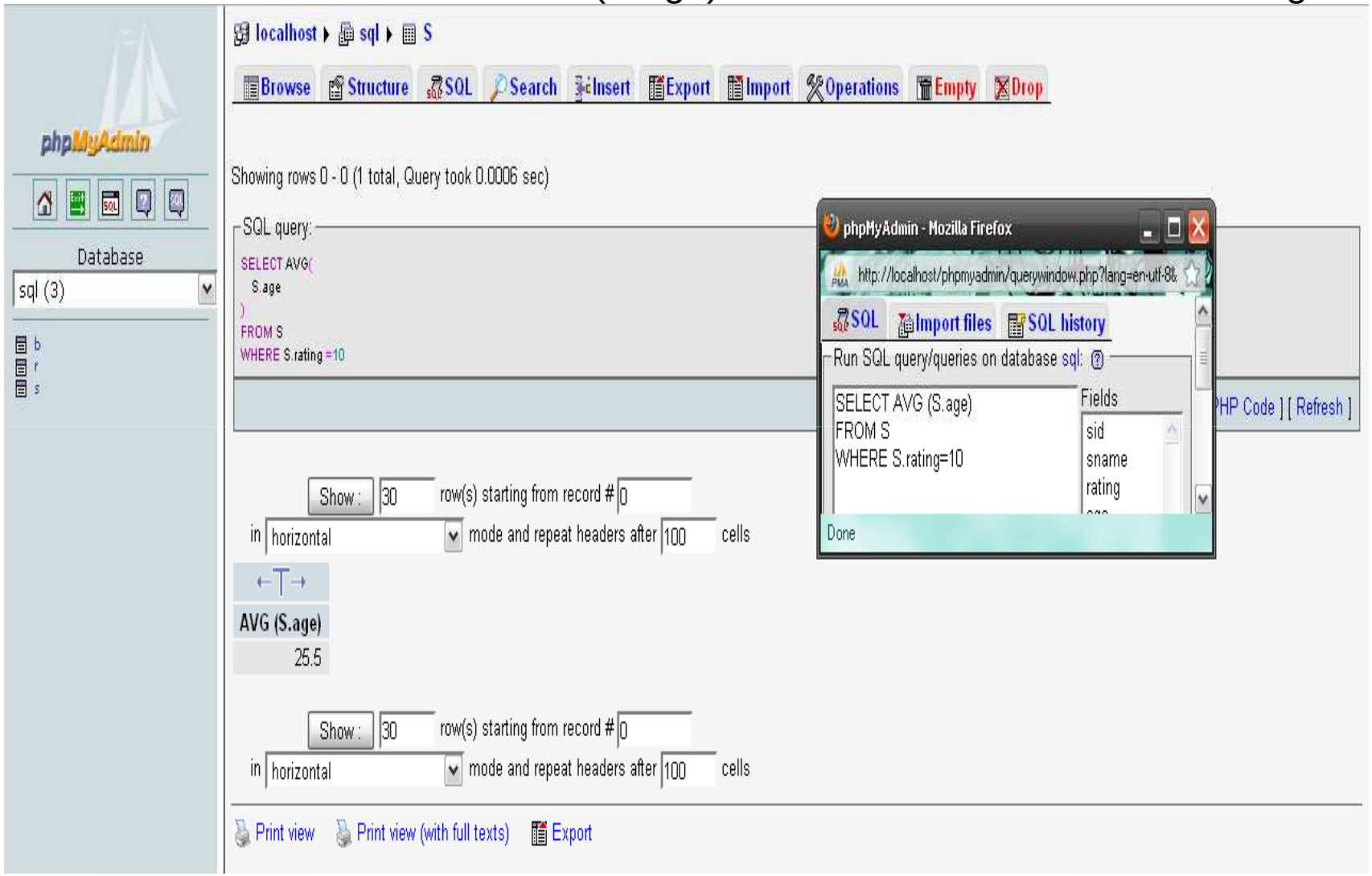
```
SELECT AVG (S.age)
        FROM Sailors S
        WHERE S.rating=10
```

```
SELECT S.sname, S.age
        FROM Sailors S
        WHERE S.age=(SELECT MAX (S2.age)
                        FROM Sailors S2)
```

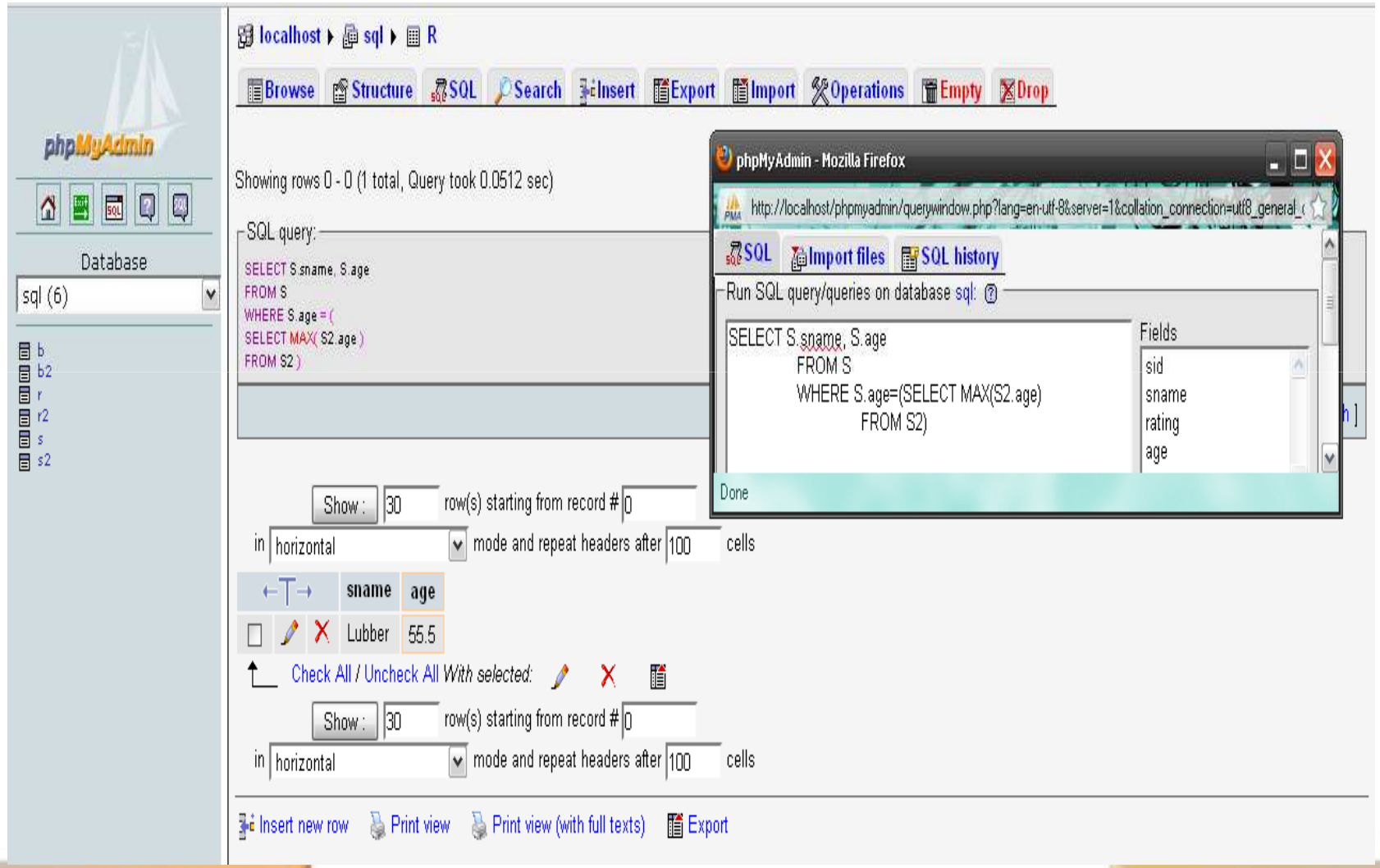
Hasil dari : SELECT COUNT (*) FROM Sailors S



Hasil dari : SELECT AVG (S.age) FROM Sailors S WHERE S.rating=10



Hasil dari : SELECT S.sname, S.age FROM Sailors S
WHERE S.age=(SELECT MAX (S2.age) FROM Sailors S2)



GROUP BY & HAVING CLAUSE

- Sejauh ini telah dibahas penggunaan operasi agregasi untuk semua tuples (yang memenuhi kualifikasi). Seringkali kita ingin menerapkan operasi agregasi untuk sejumlah kelompok (groups) dari baris (tuples) dalam suatu relasi.

- Contoh : Cari usia sailors termuda untuk setiap tingkat rating yang ada.

```
SELECT MIN(S.age)
                    FROMSailors S
                    WHERE S.rating=i
```

Jika misalnya, kita ketahui bahwa nilai rating berada dalam range 1 s.d 10, maka i pada pernyataan diatas = 1,2,3,...,10.

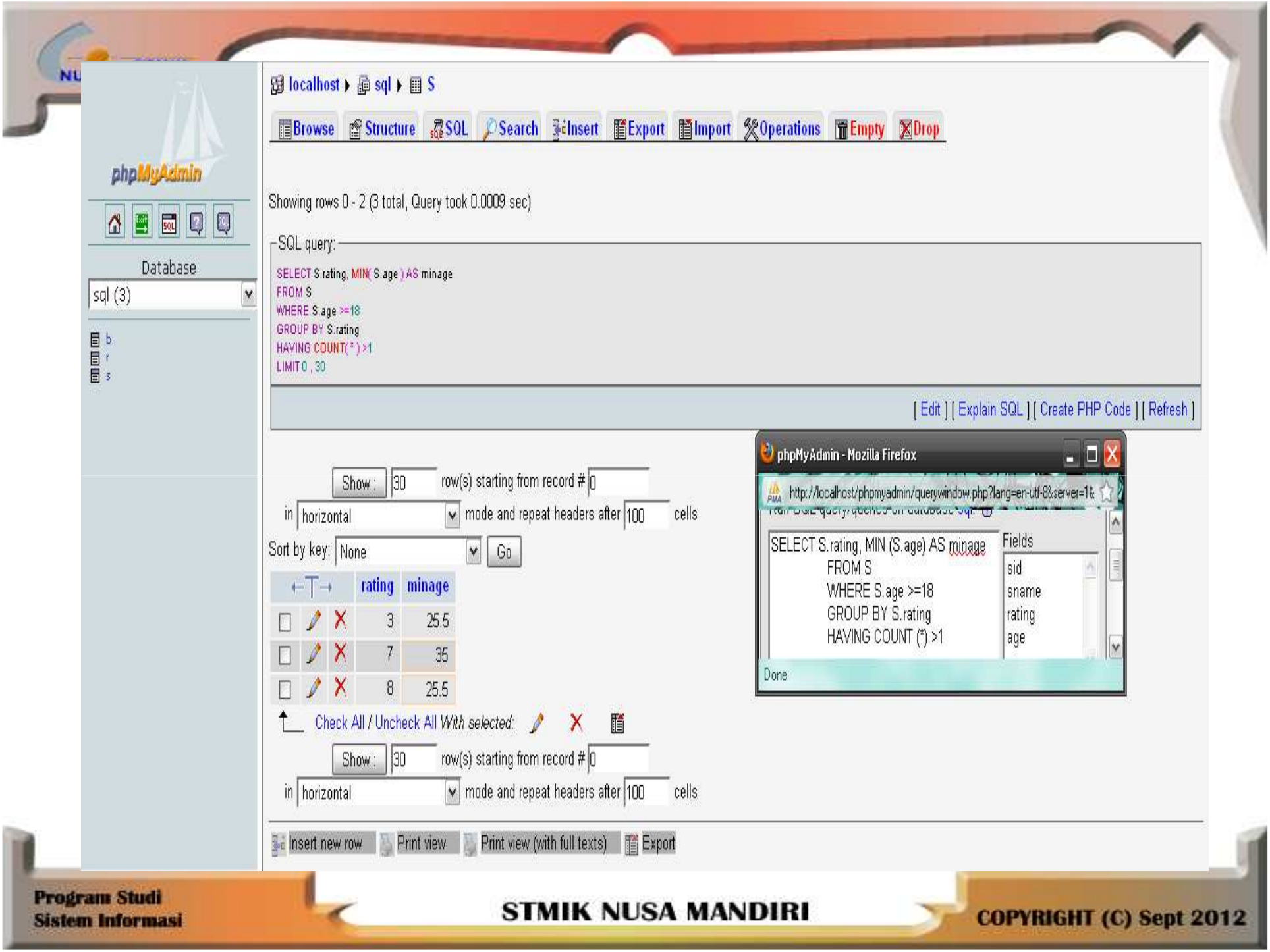
Queries dengan GROUP BY & HAVING

- Bentuk umum SQL Query dengan GROUP BY & HAVING

```
SELECT [DISTINCT] select-list
FROM from-list
WHERE qualification
GROUP BY grouping-list
HAVING group-qualification
```

- Contoh : Carilah usia sailors termuda yang memenuhi syarat (misalnya, setidaknya >=18 tahun) untuk setiap tingkat rating dengan setidaknya terdiri dari dua sailors untuk setiap tingkat ratingnya.

```
SELECT S.rating, MIN (S.age) AS minage
                 FROM Sailors S
                 WHERE S.age >=18
                 GROUP BY S.rating
                 HAVING COUNT (*) >1
```


phpMyAdmin
Database
sql (3)
b
r
s
localhost ▸ sql ▸ S
Browse
Structure
SQL
Search
Insert
Export
Import
Operations
Empty
Drop
Showing rows 0 - 2 (3 total, Query took 0.0009 sec)
SQL query:
SELECT S.rating, MIN(S.age) AS minage
FROM S
WHERE S.age >=18
GROUP BY S.rating
HAVING COUNT(*) >1
LIMIT 0 , 30
[Edit] [Explain SQL] [Create PHP Code] [Refresh]
Show : 30 row(s) starting from record # 0
in horizontal mode and repeat headers after 100 cells
Sort by key: None Go
rating minage
3 25.5
7 35
8 25.5
Check All / Uncheck All With selected:
Show : 30 row(s) starting from record # 0
in horizontal mode and repeat headers after 100 cells
Insert new row
Print view
Print view (with full texts)
Export
phpMyAdmin - Mozilla Firefox
http://localhost/phpmyadmin/querywindow.php?lang=en-utf-8&server=1&
SELECT S.rating, MIN (S.age) AS minage
FROM S
WHERE S.age >=18
GROUP BY S.rating
HAVING COUNT (*) >1
Fields
sid
sname
rating
age
Done

NULL VALUES

- Nilai-nilai fields dalam sebuah tuple kadang-kadang tidak diketahui (unknown). Misalnya : sebuah nilai rating tidak diberikan atau tidak dapat digunakan (inapplicable). Maka SQL menyediakan nilai kolom khusus yang disebut null untuk digunakan dalam situasi tersebut.

NULL VALUES dan Operator Perbandingan serta Logical Connectives AND, OR, NOT

- SQL menyediakan operator perbandingan khusus IS NULL untuk menguji apakah kolom nilai nol yang akan mengevaluasi dengan benar pada AND yang mewakili baris. Disini juga terdapat IS NOT NULL, yang akan mengevaluasi nilai false pada baris untuk AND.

Dampak NULL VALUES dalam Membangun SQL

- Untuk kualifikasi dalam klausa WHERE clause, keberadaan null values dapat menghilangkan baris (dalam garis-produk dari tabel disebutkan dalam klausa FROM) yang kualifikasi tidak mengevaluasi nilai TRUE.
  - Menghilangkan baris yang mengevaluasi unknown mempunyai dampak yang halus namun signifikan pada queries, terutama nested queries yang melibatkan EXISTS atau UNIQUE.

- Persoalan lain adalah definisi SQL yang menyatakan bahwa dua baris duplikat jika kolom yang sesuai adalah sama baik, atau keduanya bersifat null. Dalam kenyataannya jika kita membandingkan dua nilai null menggunakan =, hasilnya adalah unknown! Dalam konteks duplikat, perbandingan ini secara implisit diperlakukan sebagai nilai true, yang merupakan anomali.

- Operator aritmatika +, -, *, dan / semua menghasilkan nilai null jika salah satu dari argumennya bernilai null.

Dampak NULL VALUES dalam Membangun SQL (Lanjutan)

- Null Values dapat menimbulkan hal yang tidak diharapkan untuk operatorr-operator agegasi :

  - COUNT (*) menangani nilai null seperti halnya nilai-nilai lainnya (ikut diperhitungkan).

  - Operasi-operasi agregasi lainnya (COUNT, SUM, AVG, MIN, MAX, dan variasi penggunaan DISTINCT) hanya mengabaikan null values.

OUTER JOINS

- Beberapa varian menarik dari operasi join yang mengandalkan null values disebut **outer joins.**

- Terdapat tiga variasi outer join :
  1. Left Outer Join
  2. Right Outer Join
  3. Full Outer Join

- Sebagai contoh, query berikut adalah daftar sid, pasangan sesuai dengan pelaut dan mereka yang telah memesan perahu:

```
SELECT  Sailors.sid, Reserves.bid
FROM Sailors NATURAL LEFT OUTER JOIN Reserve R
```

phpMyAdmin

Database

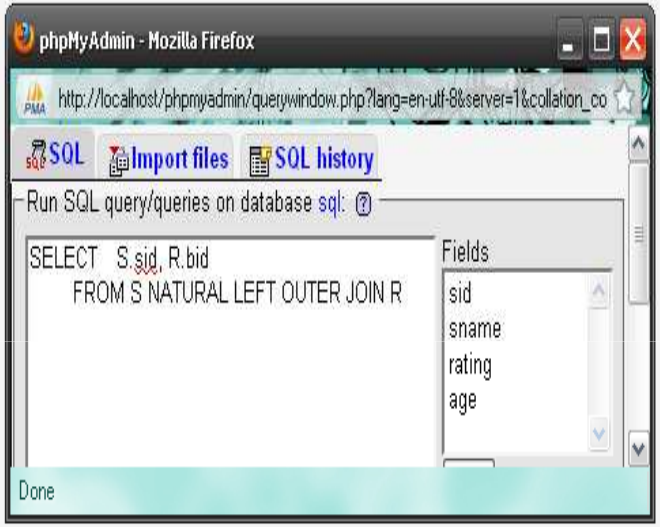
sql (3)

- b
- r
- s

Show : 30 row(s) starting from record # 0

in horizontal mode and repeat headers after 100 cells

sid	bid
22	101
22	102
22	103
22	104
29	*NULL*
31	102
31	103
31	104
32	*NULL*
58	*NULL*
64	101
64	102
71	*NULL*
74	103
85	*NULL*
95	*NULL*



Show : 30 row(s) starting from record # 0

in horizontal mode and repeat headers after 100 cells

Tidak Membolehkan Null Values

- Melarang nilai bersifat null dengan menetapkan NOT NULL sebagai bagian dari definisi sebuah field, misalnya :

  sname CHAR (20) NOT NULL

- Selain itu, field dalam PRIMARY KEY tidak diperbolehkan bernilai null. Dengan demikian, ada kendala penggunaan NOT NULL secara implisit untuk setiap field yang tercantum dalam PRIMARY KEY.