Common MySQL Queries

Extending Chapter 9 of Get it Done with MySQL 5&6

TreeView

Entries in italics are new or updated in the last month

More MySQL tips and snippets here

Basic aggregation

Aggregates are more popular than any other lookup topic here, by a ratio of more than 2 to 1.

Basic aggregation is the simplest grouping query pattern: for column *foo*, display the smallest, largest, sum, average or some other statistic of column *bar* values:

```
SELECT foo, MIN(bar) AS bar
FROM tbl
GROUP BY foo
```

Return the highest *bar* value for each *foo*, ordering top to bottom by that value:

```
SELECT foo, MAX(bar) AS Count
FROM tbl
GROUP BY foo
ORDER BY Count DESC;
```

Ditto for `AVG(), COUNT()` etc. The pattern easily extends to multiple grouping column expressions.

For aggregating functions like `MIN()` and `MAX()` that return a single value, there may be multiple instances of the result. If you wish to see them, put the aggregating query in a subquery and join to it from a direct query on the table:

```
SELECT a.foo, a.bar
FROM tbl a
JOIN (
  SELECT foo, MAX(bar) AS Count
  FROM tbl
  GROUP BY foo
) b ON a.foo=b.foo AND a.bar=b.count
ORDER BY a.foo, a.bar;
```

MySQL introduced the SQL extension `GROUP_CONCAT()`, which makes short work of listing items in groups. For example, given a table of suppliers and the parts they make ...

```
CREATE TABLE supparts(supID char(2),partID char(2));
INSERT INTO supparts VALUES
('s1','p1'),('s1','p2'),('s1','p3'),('s1','p4'),('s1','p5'),('s1','p6'),
('s2','p1'),('s2','p2'),('s3','p2'),('s4','p2'),('s4','p4'),('s4','p5');
```

list suppliers for each part:

```
SELECT partID,GROUP_CONCAT(supID ORDER BY supID) AS Suppliers
FROM supparts
GROUP BY partID;
+--------+-------------+
| partID | Suppliers   |
+--------+-------------+
| p1     | s1,s2       |
| p2     | s1,s2,s3,s4 |
| p3     | s1          |
| p4     | s1,s4       |
| p5     | s1,s4       |
| p6     | s1          |
+--------+-------------+
```

When there are several tables to be joined, the beginner may feel overwhelmed by the complexity of the problem. Suppose you're asked to retrieve the top computer desk salesperson for this schema:

```
drop table if exists salespersons, orders, orderlines, products;
create table salespersons(salespersonid int,name char(8));
insert into salespersons values(1,'Sam'),(2,'Xavier');
create table orders(orderid int,salespersonid int);
insert into orders values(1,1),(2,1),(3,1),(4,2),(5,2);
create table orderlines(lineid int,orderid int,productid int,qty int);
insert into orderlines values(1,1,1,1),(2,1,1,2),(3,2,2,1),(4,3,1,1),(5,4,1,1),
(6,5,2,2);
create table products(productid int,name char(32));
insert into products values(1,'computer desk'),(2,'lamp'),(3,'desk chair');
```

Two rules of thumb help with problems like this: solve one step at a time, and work from inside out. Here "inside out" means start by *building the join list* needed to retrieve sales data:

```
from salespersons s
join orders       o using(salespersonid)
join orderlines   l using(orderid)
```

```
join products     p using(productid)
```

Test those joins with a query that just lists sales data:

```
select s.name, p.name, l.qty
from salespersons s
join orders       o using(salespersonid)
join orderlines   l using(orderid)
join products     p using(productid)
+--------+---------------+------+
| name   | name          | qty  |
+--------+---------------+------+
| Sam    | computer desk |    1 |
| Sam    | computer desk |    2 |
| Sam    | lamp          |    1 |
| Sam    | computer desk |    1 |
| Xavier | computer desk |    1 |
| Xavier | lamp          |    2 |
+--------+---------------+------+
```

Now we just need to filter for 'computer desk' sales, add aggregation, and pick off the top seller:

```
select s.name, sum(l.qty) as n      -- sum quantities
from salespersons s
join orders       o using(salespersonid)
join orderlines   l using(orderid)
join products     p using(productid)
where p.name='computer desk'
group by s.name                     -- aggregate by salesperson
order by n desc limit 1;            -
- order by descending sum, pick off top value
+------+------+
| name | n    |
+------+------+
| Sam  |    4 |
+------+------+
```

If columns other than the `GROUP BY` column must be retrieved, and if the grouping expression does not have a strictly 1:1 relationship with those columns, then to avoid returning arbitrary values for those non-grouping columns, you must put the `GROUP BY` query in a subquery and join that result to the other columns, for example:

```
SELECT s.partID, s, thiscol, s.thatcol, anothercol, x.Suppliers
FROM supparts s
JOIN (
  SELECT partID,GROUP_CONCAT(supID ORDER BY supID) AS Suppliers
  FROM supparts
  GROUP BY partID
) x USING(partID)
```

Last updated 05 Jan 2013

Back to the top *Browse the book* *Buy the book* *Feedback*

Aggregate across columns

You track squash court bookings and fees. A court booking has one fee, but it has two member references, and those can be the same if one member has brought a guest. For each booking row, the fee is to be split between the two members. What query correctly aggregates fees including cases where the two members of record are the same?

```
DROP TABLE IF EXISTS bookings;
CREATE TABLE bookings (
  court_id int(11) NOT NULL,
  member1 int(11) NOT NULL,
  member2 int(11) NOT NULL,
  time timestamp NOT NULL,
  fee decimal(5,2) NOT NULL
);

INSERT INTO bookings ( court_id , member1 , member2 , time , fee )
VALUES
(1, 1000, 1001, '2009-09-09 15:49:38', 3.00),
```

```
(2, 1000, 1000, '2009-09-08 15:50:04', 3.00);
```

For this data the correct result is

```
member fees
1000   4.50
1001   1.50
```

An efficient solution, posted by 'laptop alias' on a MySQL forum:

```
SELECT member, ROUND(SUM(fee/2),2) AS total
FROM (
  SELECT member1 AS member, fee FROM bookings
  UNION ALL
  SELECT member2, fee FROM bookings
) AS tmp
GROUP BY member;
```

Last updated 09 Sep 2009

Back to the top | Browse the book | Buy the book | Feedback

Aggregates across multiple joins

Given a parent table and two child tables, a query which sums values in both child tables, grouping on a parent table column, returns sums that are exactly twice as large as they should be. In this example from the MySQL General Discussion list:

```
DROP TABLE IF EXISTS packageItem,packageCredit,packageItemTax;
CREATE TABLE packageItem (
 packageItemID INT,
 packageItemName CHAR(20),
 packageItemPrice DECIMAL(10,2)
);
INSERT INTO packageItem VALUES(1,'Delta Hotel',100.00);

CREATE TABLE packageCredit (
 packageCreditID INT,
 packageCreditItemID INT,
```

Read the entire item

Last updated 22 Feb 2013

Back to the top | Browse the book | Buy the book | Feedback

Aggregates excluding leaders

You have a table of grouped ranks ...

```
DROP TABLE IF EXISTS grps,ranks;
CREATE TABLE grps (grp int);
INSERT INTO grps VALUES(1),(2),(3),(4);
CREATE TABLE ranks(grp int,rank int);
INSERT INTO ranks VALUES(1, 4 ),(1, 7 ),(1, 9 ),(2, 2 ),(2, 3 ),(2, 5 ),(2, 6 ),
(2, 8 ),(3, 1 ),(4,11 ),(4,12 ),(4,13 );
```

and you wish to list ranks by group *omitting the leading rank in each group*. The simplest query for group leaders is ...

Read the entire item

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Aggregates of specified size

Find the values of a table column c1 for which there are a specified number of listed values in another column c2.

To get an overview of the values of c2 for each value of c1:

```
SELECT
  c1,
  GROUP_CONCAT(c2 ORDER BY c2) AS 'C2 values'
FROM table
GROUP BY c1;
```

To retrieve a list of c1 values for which there exist specific values in another column c2, you need an IN clause specifying the c2 values and a HAVING clause specifying the required number of different items in the list ...

```
SELECT c1
FROM table
WHERE c2 IN (1,2,3,4)
GROUP BY c1
HAVING COUNT(DISTINCT c2)=4;
```

This is easy to generalise to multiple column expressions, and a HAVING clause specifying any number of items from the IN list.

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

All X for which all Y are Z

You have an election database with tables for candidates, parties and districts. A candidate belongs to one party; a district may have any number of candidates:

```
DROP TABLE IF EXISTS parties,districts,candidates;
CREATE TABLE parties (
  party char(12) NOT NULL,
  PRIMARY KEY (party)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
INSERT INTO parties VALUES ('Conservative'),('Liberal'),('Socialist'),('Green'),
('Libertarian');

CREATE TABLE districts (
  district char(10) DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
INSERT INTO districts VALUES ('Essex'),('Malton'),('Riverdale'),('Guelph'),
('Halton');
```

Read the entire item

Last updated 19 Mar 2016

Back to the top | Browse the book | Buy the book | Feedback

All X meeting multi-row conditions

Given this table ...

```
drop table if exists mpu;
create table mpu( id int, id_marketplace int, label int, property int);
insert into mpu values
(1 , 10 , 3 , 0 ),(2 , 10 , 6 , 35 ),(4 , 10 , 10 , 22 ),(5 , 10 , 9 , 0 ),
(6 , 11 , 3 , 0 ),(7 , 11 , 6 , 5 ),(8 , 11 , 7 , 7 ),(9 , 11 , 7 , 10 ),
(10 , 11 , 10 , 21),(11 , 12 , 3 , 0 ),(12 , 12 , 6 , 5 ),(13 , 12 , 7 , 8 ),
(14 , 12 , 7 , 9 ),(15 , 12 , 10 , 21 ),(16 , 13 , 3 , 0  ),(17 , 13 , 6 , 35 ),
(18 , 13 , 7 , 7),(19 , 13 , 7 , 8 ),(20 , 13 , 10 , 20 );select * from mpu;
+------+----------------+-------+----------+
| id   | id_marketplace | label | property |
+------+----------------+-------+----------+
```

Read the entire item

Last updated 19 Jan 2014

Back to the top | Browse the book | Buy the book | Feedback

Avoiding repeat aggregation

In a good introductory tutorial on MySQL subqueries, Jeremy Cole developed a triply nested query to retrieve the name, population, head of state, and number of official languages in countries with the most official languages. The query uses two tables from the MySQL `world` database:

```
CREATE TABLE country (
  Code char(3) NOT NULL DEFAULT '' PRIMARY KEY,
  Name char(52) NOT NULL DEFAULT '',
  Population int(11) NOT NULL DEFAULT '0',
  HeadOfState char(60) DEFAULT NULL,
  ... other columns ...
);
CREATE TABLE countrylanguage (
  CountryCode char(3) NOT NULL DEFAULT '' PRIMARY KEY,
  Language char(30) NOT NULL DEFAULT '',
  IsOfficial enum('T','F') NOT NULL DEFAULT 'F',
```

Read the entire item

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Break out sum on condition

To obtain part sums of a column based on specific conditions, use If() or `CASE`, eg ...

```
select
  sum( if(condition1, col, 0) ) as cond1,
  sum( if(condition2, col, 0) ) as cond2,
  ...etc...
from tbl;
```

Last updated 16 Aug 2012

Back to the top | Browse the book | Buy the book | Feedback

Cascading aggregates

When you have parent-child-grandchild tables, eg `companies, users, actions`, and your query requirement is for per-parent aggregates from the child table and per-child aggregates from the grandchild table, then cascading joins yield spuriously multiplied counts, and correlated subqueries fail because the second correlated subquery cannot find a visible joining column.

One solution is to use derived tables. Assuming ...

```
DROP TABLE IF EXISTS companies,users,actions;
CREATE TABLE companies (id int, name char(10));
INSERT INTO COMPANIES VALUES(1,'abc ltd'),(2,'xyz inc');
CREATE TABLE users (id INT,companyid INT);
INSERT INTO users VALUES(1,1),(2,1),(3,1),(4,2),(5,2);
CREATE TABLE actions (id INT, userid INT, date DATE);
INSERT INTO actions VALUES
( 1, 1, '2009-1-2'),( 2, 1, '2009-1-3'),( 3, 2, '2009-1-4'),( 4, 2, '2009-1-5'),
( 5, 3, '2009-1-6'),
```

Read the entire item

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Combine Group_Concat() with counts

Rather than list instances including repeats on one line, you want to list distinct instances and their counts. One way is to do a simple `GROUP BY` query and in your

application layer remove the newlines from the result. Or you can do it in one step:

```
drop table if exists t;
create table t (
  type int(10) ,
  instance int(10)
) ;
insert into t values
(1,4),(1,7),(1,9),(1,10),(2,2),(2,3),(2,5),(2,6),(2,8),(3,1),(4,11);

select group_concat( concat( type,'(',qty,')') separator ', ') list
from (
  select type, count(*) qty
  from t
  group by type
) n
+------------------------+
| list                   |
+------------------------+
| 1(4), 2(5), 3(1), 4(1) |
+------------------------+
```

Last updated 28 Jun 2012

Back to the top *Browse the book* *Buy the book* *Feedback*

Correlated aggregating subqueries

If you need a child table aggregate result for each row in a query, what's faster, putting the correlated aggregating subquery in the `SELECT` list, or in the `FROM` clause?

Thus, given tables `stocks(id, user, ticker,lastvalue)` and `transactions(id, sid, action, qty, price, commission)`, and the requirement to retrieve stock values and per-stock transaction counts, which of these queries will run faster?

```
select ticker, id, lastvalue, (select count(*) from transactions where sid=stocks.id) N
from stocks;
```

... or ...

```
select s.ticker, s.id, s.lastvalue, t.N
from stocks s
join (
  select sid, count(*) N
  from transactions
  group by sid
) t on s.id=t.sid;
```

The first query's syntax is simpler, so it's often the first choice. `EXPLAIN` reports that the second query requires examination of more rows than the first query does.

But benchmarks with caching turned off show that the *second* query is much faster--mainly because it executes *one* subquery rather than a subquery per row.

If you have a choice, put the aggregating subquery in the `FROM` clause.

Last updated 10 Oct 2014

Back to the top *Browse the book* *Buy the book* *Feedback*

Cross-aggregates

Given the table authorbook(authid INT, bookid INT), what query finds the books who have authors with more than one book in the table?

Even one level of recursion can induce a mild trance. Escape the trance by taking the problem one step at a time. First write the query that finds the authors with multiple books. Then join an outer query to that on authorid, and have the outer query select bookid:

```
SELECT a1.bookid
FROM authorbook a1
INNER JOIN (
  SELECT authid,count(bookid)
  FROM authorbook a2
  GROUP BY authid
  HAVING COUNT(bookid)>1
) AS a3 ON a1.authid=a3.authid;
```

Last updated 22 May 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

Every paying customer wins a prize

A superstore is running a promotion: each day, every fifth a customer wins a prize. Each day you're given a text file with data columns for customerID, timestamp and order amount. How do you find every fifth customer?

Load the data into a table, then write the query inside out; with this kind of problem, nested queries are a boon.

1. Using Load Data Infile, load the text file into a table (indexless for speed) named `tbl` with columns for customerID, timestamp and amount, and index the table on (customerID, timestamp).

2. Write a query that tracks per-customer order by timestamp with user variables:

```
set @id='', @ord=1;
select
```

Read the entire item

Last updated 12 Sep 2013

Back to the top *Browse the book* *Buy the book* *Feedback*

Group data by datetime periods

To group rows by a time period whose length in *minutes* divides evenly into 60, use this formula:

```
GROUP BY ((60/periodMinutes) * HOUR( thistime ) + FLOOR( MINUTE( thistime ) / periodMinutes ))
```

where `thistime` is the `TIME` column and `periodMinutes` is the period length in minutes. So to group by 15-min periods, write ...

```
SELECT ...
GROUP BY ( 4 * HOUR( thistime ) + FLOOR( MINUTE( thistime ) / 15 ))
...
```

Read the entire item

Last updated 22 May 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

League table

Here is a simple soccer league table setup that was developed in the MySQL Forum by J Williams and a contributor named "Laptop Alias". The `teams` table tracks team ID and name, the `games` table tracks home and away team IDs and goal totals for each game. The query for standings is built by aggregating a `UNION` of home team and away team game results:

```
DROP TABLE IF EXISTS teams, games;
CREATE TABLE teams(id int primary key auto_increment,tname char(32));
CREATE TABLE games(id int primary key auto_increment, date datetime,
```

```
                    hteam int, ateam int, hscore tinyint,ascore tinyint);

INSERT INTO teams VALUES(1,'Wanderers'),(2,'Spurs'),(3,'Celtics'),(4,'Saxons');
INSERT INTO games VALUES
(1,'2008-1-1 20:00:00',1,2,1,0),(2,'2008-1-1 20:00:00',3,4,0,2),
(3,'2008-1-8 20:00:00',1,3,1,1),(4,'2008-1-8 20:00:00',2,4,2,1);
SELECT * FROM teams;
+----+-----------+
```

Read the entire item

Last updated 22 May 2009

Back to the top Browse the book Buy the book Feedback

Nested, banded aggregation

Employee sales commission rates increase with sales totals according to specified bands of sales total amounts—like a graduated income tax in reverse. To compute total commissions due each employee, we need to aggregate twice: first to get sales per employee, then to get commissions per employee:

```
DROP TABLE IF EXISTS sales, commissions;
CREATE TABLE sales(employeeID int,sales int);
INSERT INTO sales VALUES(1,2),(1,5),(1,7),(2,9),(2,15),(2,12);
SELECT * FROM sales;
+------------+-------+
| employeeID | sales |
+------------+-------+
|          1 |     2 |
|          1 |     5 |
|          1 |     7 |
|          2 |     9 |
```

Read the entire item

Last updated 15 Nov 2009

Back to the top Browse the book Buy the book Feedback

Pairwise matchmaking

Given tables tracking users and their hobbies, how do we write a query that ranks pairs of users on hobby similarity?

```
DROP TABLE IF EXISTS users,hobbies,users_hobbies;
CREATE TABLE users( id int, name char(16) ) ;
INSERT INTO users VALUES (1,'John'),(2,'Lewis'),(3,'Muhammad');
CREATE TABLE hobbies( id int, title char(16) ) ;
INSERT INTO hobbies
VALUES (1,'Sports'),(2,'Computing'),(3,'Drinking'),(4,'Racing'),(5,'Swimming'),
(6,'Photography');
CREATE TABLE users_hobbies( user_id int, hobby_id int ) ;
INSERT INTO users_hobbies
VALUES (1,2),(1,3),(1,6),(2,1),(2,5),(2,6),(3,2),(3,5),(3,6),(1,2),(1,3),(1,6),
(2,1),
(2,5),(2,6),(3,2),(3,5),(3,6),(1,2),(1,3),(1,6),(2,1),(2,5),(2,6),(3,2),(3,5),
(3,6);
```

Read the entire item

Last updated 17 Sep 2010

Back to the top Browse the book Buy the book Feedback

Show only one child row per parent row

Given tables `parent(id int not null primary key, etc...)` and `child (id int not null primary key, pid int not null references parent (id), etc...)`, how do we write a query that retrieves only one child row per `pid` even when the child table has multiple matching rows? MySQL permits use of `GROUP BY` even when the `SELECT`

list specifies no aggregate function, so this will work:

```
select p.id, c.id
from parent p
join child c on p.id=c.pid
group by p.id;
```

But is it accurate? No, because it displays only the first c.pid value it happens to find. For further discussion see Within-group aggregates.

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Skip repeating values

You want to report all unique values of a column and skip all rows repeating any of these values.

```
SELECT col
FROM foo
GROUP BY col
```

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Totals and subtotals--simply

You have a table that tracks attendance hours ...

```
drop table if exists t;
create table t(d date, id int, hrs int);
insert into t values
('2013-10-1',1,5), ('2013-10-1',2,6), ('2013-10-1',3,2), ('2013-10-1',3,5),
('2013-10-2',1,1), ('2013-10-2',1,2), ('2013-10-2',2,3), ('2013-10-2',2,4),
('2013-10-3',1,2), ('2013-10-3',1,2), ('2013-10-3',1,2);
```

... and you need an attendance summary by date and person. It turns out that this aggregating query is easy top write—first write an inner query to Group By date and person With Rollup, then write an outer query that renames the Rollup Null labels.

Read the entire item

Last updated 31 Oct 2013

Back to the top | Browse the book | Buy the book | Feedback

Track state changes

You have a table containing a sequence of IDs, states and dates. The query requirement is to list state changes and their dates ordered by date, and count instances of each sequenced state. This data from a blog entry by a Russian MySQLer calling himself Quassnoi ...

```
+----+-------+------------------+
|ID  | State |  Datetime        |
+----+-------+------------------+
| 12 |   1   | 2009-07-16 10:00 |
| 45 |   2   | 2009-07-16 13:00 |
| 67 |   2   | 2009-07-16 14:40 |
| 77 |   1   | 2009-07-16 15:00 |
| 89 |   1   | 2009-07-16 15:30 |
| 99 |   1   | 2009-07-16 16:00 |
+----+-------+------------------+
```

Read the entire item

Last updated 20 Oct 2014

Back to the top | Browse the book | Buy the book | Feedback

Values linked with all values of another column

You have a table in which each row references one text and one keyword in the text ...

```
DROP TABLE IF EXISTS keywords;
CREATE TABLE keywords (txtID int, keyword char(8));
INSERT INTO keywords VALUES(1 , 'foo'),(2 , 'bar'),(1 , 'foo'),(2 , 'foo');
```

... and you want a list of texts which include every keyword.

You might think you have to join and match. You don't. All you need to do is count the distinct keywords which occur for each text, then for each text compare that number with the entire list of distinct keywords:

```
SELECT txtID, COUNT(DISTINCT keyword) AS N
FROM keywords
GROUP BY txtID
HAVING N = (SELECT COUNT(DISTINCT keyword) FROM keywords);
+-------+---+
| txtID | N |
+-------+---+
|     2 | 2 |
+-------+---+
```

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Within-group aggregates

You have a `products` table with columns `item, supplier, price`. Multiple suppliers offer various prices for the same item. You need to find the supplier with the lowest price for each item.

```
DROP TABLE IF EXISTS products;
CREATE TABLE products(item int,supplier int,price decimal(6,2));
INSERT INTO products VALUES(1,1,10),(1,2,15),(2,2,20),(2,1,21),(2,2,18);
SELECT * FROM products;
+------+----------+-------+
| item | supplier | price |
+------+----------+-------+
|    1 |        1 | 10.00 |
|    1 |        2 | 15.00 |
|    2 |        2 | 20.00 |
|    2 |        1 | 21.00 |
|    2 |        2 | 18.00 |
+------+----------+-------+
```

Your first thought may be to `GROUP BY item`, but that is not guaranteed to return the correct `supplier` value for each minimum item price. Grouping by both `item` and `supplier` will return more information than you want. Nor can you write `WHERE price=MIN(...)` because the query engine will evaluate the `WHERE` clause before it knows the `MIN` value.

This is the problem of *aggregating within aggregates*. It is sometimes called the 'groupwise aggregates' problem, but the term 'groupwise' is ambiguous. We think better names for it are *subaggregates*, *inner aggregates*, or *within-group aggregates*.

It's easy to show that the within-group aggregates problem is a form of the problem of returning values from non-grouping columns in a `GROUP BY` query. Suppose you write ...

```
SELECT item, supplier, Min(price)
FROM products
GROUP BY item;
```

Will this reliably return the correct supplier per item? *No. Unless there is exactly one supplier per item, the supplier value returned will be arbitrary*. To retrieve the correct supplier for each item, you need more query logic.

Here are six solutions. As you'll see, some are very slow, a couple are quite fast, and some work for only some versions of the problem:

1. Self-exclusion join solution: One way to model within-group minima or maxima is via an *left self exclusion join*...

```
SELECT p1.item, p1.supplier, p1.price
FROM products AS p1
LEFT JOIN products AS p2 ON p1.item  = p2.item AND p1.price > p2.price
WHERE p2.item IS NULL;
```

...because in the resultset built by joining on left item = right item and left price > right price, the left-sided rows for which there is no greater right-sided price are precisely the per-item rows with the smallest prices. The query runs more than an order of magnitude faster with an index on `(item, supplier)`.

2. Intermediate table solution: Another solution is to derive an intermediate table of aggregated minimum prices, then query that table. Before MySQL 4.1, the intermediate table has to be a temporary table:

```
CREATE TEMPORARY TABLE tmp (
  item INT,
  minprice DECIMAL DEFAULT 0.0
);
LOCK TABLES products READ;
INSERT INTO tmp
  SELECT item, MIN(price)
  FROM products
  GROUP BY item;
```

to which you then join the products table to find the matching suppliers:

```
SELECT products.item, supplier, products.price
FROM products
JOIN tmp ON products.item = tmp.item
WHERE products.price=tmp.minprice;
UNLOCK TABLES;
DROP TABLE tmp;
```

3. Correlated subquery solution: From MySQL 4.1 on, the temporary table can be *a correlated subquery*. This is the most intuitively obvious syntax for the problem. It's also the *slowest*—up to a hundred times slower than the exclusion join, whether the queries are compared with or without indexes:

```
SELECT item, supplier, price
FROM products AS p1
WHERE price = (
  SELECT MIN(p2.price)
  FROM products AS p2
  WHERE p1.item = p2.item
);
```

4. `FROM` clause subquery solution: If we move the aggregating subquery from the `WHERE` clause to the `FROM` clause, the query improves from a hundred times slower than the self-exclusion join to *twenty times faster*:

```
SELECT p.item, p.supplier, p.price
FROM products AS p
JOIN (
  SELECT item, MIN(price) AS minprice
  FROM products
  GROUP BY item
) AS pm ON p.item = pm.item AND p.price = pm.minprice;
```

Some users have trouble mapping elements of this model to their instance of the problem. The model has five elements (or sets of them):

(i) a table, which might be a view, a single physical table, or a table derived from joins
(ii) one or more grouping columns,

(iii) one or more columns to aggregate,
(iv) one or more columns not mentioned in the `GROUP BY` clause,
(v) an aggregating job to do, typically MIN() or MAX().

In the product/minimum price solution above:

(i) table `tbl = product`
(ii) grouping column `grouping_col = item`
(iii) column to aggregate = `col_to_aggregate = price`
(iv) non-aggregated columns `other_detail, ...etc... = supplier`
(v) aggregating function = `MIN()`.

Here's an interesting variation on the problem. A simple table tracks company branches and the kinds of stock they carry:

```
DROP TABLE IF EXISTS branch_stock;
CREATE TABLE branch_stock(
  stock_id INT PRIMARY KEY AUTO_INCREMENT KEY,
  branch_id INT NOT NULL,
  stock_item VARCHAR(12) NOT NULL
);
INSERT INTO branch_stock (branch_id,stock_item)
VALUES (1,'trainers'),(1,'trainers'),(1,'jumper'),(2,'tie'),(2,'shoes');
```

How do we find the most frequent product for each branch, including ties? Join the intermediate table of grouped counts to itself on matching branches, non-matching stock and not-greater counts in one than in the other:

```
SELECT DISTINCT a.*
FROM (
  SELECT branch_id, stock_item, COUNT(*) qty
  FROM branch_stock
  GROUP BY branch_id, stock_item
) a
JOIN (
  SELECT branch_id, stock_item, COUNT(*) qty
  FROM branch_stock
  GROUP BY branch_id, stock_item
) b ON b.branch_id=a.branch_id AND b.stock_item<>a.stock_item AND b.qty<=a.qty;
+-----------+------------+-----+
| branch_id | stock_item | qty |
+-----------+------------+-----+
|         1 | trainers   |   2 |
|         2 | shoes      |   1 |
|         2 | tie        |   1 |
+-----------+------------+-----+
```

5. Ordered subquery solution: A "trick" solution, made possible by MySQL's non-standard tolerance of `GROUP BY` when there is no aggregating `SELECT` expression, uses `ORDER BY` in a subquery to find the lowest prices, and `GROUP BY` in the outer query to pick them off:

```
SELECT *
FROM (
  SELECT *
  FROM products
  ORDER BY price ASC
) AS s
GROUP BY item;
```

It's succinct, it's fast if the query engine can find an index to order on, and it retrieves the prices and associated values in one step, but this query is seldom much faster than the `FROM` clause subquery described above.

The self-exclusion join and `WHERE` clause subquery methods scale badly because they're $O(N^2)$. If ordering and grouping columns are indexed, they become O(N * log N), but are still substantially slower than the `FROM` clause subquery method.

6. Min-Concat-trick solution: Finally, here is a radically different model of the problem. It can find both within-group minima and within-group maxima in a single query. The model aggregates the concatenated within-group grouped column value and the within-group grouping column name in a single string, then uses `SUBSTR()` to break them apart in the result:

```
SELECT
  item,
  SUBSTR( MIN( CONCAT( LPAD(price,6,0),supplier) ), 7)   AS MinSupplier,
    LEFT( MIN( CONCAT( LPAD(price,6,0),supplier) ), 6)+0 AS MinPrice,
  SUBSTR( MAX( CONCAT( LPAD(price,6,0),supplier) ), 7)   AS MaxSupplier,
    LEFT( MAX( CONCAT( LPAD(price,6,0),supplier) ), 6)+0 AS MaxPrice
FROM  products
GROUP BY item;
+------+-------------+----------+-------------+----------+
| item | MinSupplier | MinPrice | MaxSupplier | MaxPrice |
+------+-------------+----------+-------------+----------+
|    1 | 1           |       10 | 2           |       15 |
|    2 | 2           |       18 | 1           |       21 |
+------+-------------+----------+-------------+----------+
```

Try all solutions to find which is fastest for your version of the problem.

To find the top or bottom N per group, you might think the `LIMIT` clause would work, but `LIMIT` is limited in subqueries. See Within-group quotas.

Last updated 03 Dec 2012

Back to the top *Browse the book* *Buy the book* *Feedback*

Within-group aggregates with a wrinkle

We have a wages table holding wage rates by waiter and startdate, and a tips table which tracks hours worked and tips received per waiter per day. The requirement is to report wages and concurrent tips per waiter per day.

```
DROP TABLE IF EXISTS wages,tips;
CREATE TABLE wages( id int, waiter int, start date, rate decimal(6,2));
INSERT INTO wages VALUES
( 1, 4, '2005-01-01', 5.00 ),
( 2, 4, '2005-03-01', 6.00 ),
( 3, 5, '2007-01-05', 7.00 ),
( 4, 5, '2008-03-20', 8.00 ),
( 5, 5, '2008-04-01', 9.00 );
CREATE TABLE tips(
  id int,
  date date,
```

Read the entire item

Last updated 22 May 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

Within-group quotas (Top N per group)

A table has multiple rows per key value, and you need to retrieve, say, the first or earliest two rows per key. For example:

```
DROP TABLE IF EXISTS test;
CREATE TABLE test( id INT, entrydate DATE );
INSERT INTO test VALUES
( 1, '2007-5-01' ),( 1, '2007-5-02' ),( 1, '2007-5-03' ),( 1, '2007-5-04' ),
( 1, '2007-5-05' ),( 1, '2007-5-06' ),( 2, '2007-6-01' ),( 2, '2007-6-02' ),
( 2, '2007-6-03' ),( 2, '2007-6-04' ),( 3, '2007-7-01' ),( 3, '2007-7-02' ),
( 3, '2007-7-03' );
```

One approach is to rank rows with user variables and pick off the top two for each key in the `WHERE` clause:

Read the entire item

Last updated 04 May 2010

Back to the top *Browse the book* *Buy the book* *Feedback*

Average the top 50% values per group

Each row of a games table records one game score for a team:

```
DROP TABLE IF EXISTS games;
CREATE TABLE games(id INT, teamID INT, score INT);
INSERT INTO games VALUES
  (1,1,3),(2,1,4),(3,1,5),(4,1,6),(5,2,6),
  (6,2,7),(7,2,8),(8,2,7),(9,2,6),(10,2,7);
```

How would we write a query that returns the average of the top 50% of scores per team?

Read the entire item

Last updated 22 May 2009

Back to the top Browse the book Buy the book Feedback

Correlation

Probably more bunk has been written about correlation than about any other statistic. We'll keep this short and straight. At its simplest, correlation is a statistical measure of non-random, linear association between pairs of values in a dataset. It's denoted by *r*, and varies from -1 through +1, where -1 indicates perfect inverse correlation (the regression line goes down left to right), 0 indicates no correlation (there is no regression line; it's just a scatterplot), and +1 indicates perfect direct correlation (the regression line goes up left to right).

For an example we'll use a bit of imaginary data:

```
drop table if exists t;
create table t (id int, x int, y float);
insert into t values
(1 , 68, 4.1),(2 , 71, 4.6),(3 , 62, 3.8),(4 , 75, 4.4),(5 , 58, 3.2),
(6 , 60, 3.1),(7 , 67, 3.8),(8 , 68, 4.1),(9 , 71, 4.3),(10, 69, 3.7),
(11, 68, 3.5),(12, 67, 3.2),(13, 63, 3.7),(14, 62, 3.3),(15, 60, 3.4),
(16, 63, 4.0),(17, 65, 4.1),(18, 67, 3.8),(19, 63, 3.4),(20, 61, 3.6);
```

Read the entire item

Last updated 17 Jun 2012

Back to the top Browse the book Buy the book Feedback

Count unique values of one column

```
SELECT col_name, COUNT(*) AS frequency
FROM tbl_name
GROUP by col_name
ORDER BY frequency DESC;
```

Last updated 22 May 2009

Back to the top Browse the book Buy the book Feedback

Median

Statistically, the median is the middle value--the value that is smaller than that found in half of all remaining rows, and larger than that found in the other half:

```
SELECT a.hours As Median
FROM BulbLife As a, bulbLife AS b
GROUP BY a.Hours
Having Sum( Sign( 1-Sign(b.Hours-a.Hours ))) = Floor(( Count(*)+1)/2)  ;
```

This formula works, but it doesn't scale—it's O(n*n)—and it's awkward to use.

So we posted a MySQL implementation of Torben Mogenson's algorithm for

calculating the median (http://ndevilla.free.fr/median/median/node20.html). It's said to be amongst the fastest, but it proved too slow. Then Joe Wynne offered an algorithm which looks correct and does scale. Here it is as a MySQL stored procedure:

Read the entire item

Last updated 12 Dec 2012

Back to the top | Browse the book | Buy the book | Feedback

Mode

Statistically, the mode is the most frequently occurring value. Given tables parent(id int) and child(pid int, cid int), where child.pid references parent.id as a foreign key, what query finds the parent.id most often represented in the child id, that is, the modal count of child.pid?

```
SELECT pid, COUNT(*) AS frequency
FROM child
GROUP BY pid
ORDER BY frequency DESC
LIMIT 1;
```

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Rank order

Without MSSQL's `RANK()` aggregate function, how do we display rank order in a MySQL query, for example from a table like this?

```
CREATE TABLE votes( name CHAR(10), votes INT );
INSERT INTO votes VALUES
  ('Smith',10),('Jones',15),('White',20),('Black',40),('Green',50),('Brown',20);
```

The query is a two-step:
1. Join the table to itself on the value to be ranked, handling ties
2. Group and order the result of the self-join on rank:

Read the entire item

Last updated 24 Apr 2014

Back to the top | Browse the book | Buy the book | Feedback

Aggregates from bands of values

Aggregating by bands of values can clarify data patterns. The banding trick is a simple transformation on the banding column:

```
<band width> * Floor( <banding column> / <band width> )
```

so to count and average scores in bands of 10, ie 0-9,10-19 and so on ...

```
create table scores(score int);
insert into scores values(5),(15),(25),(35);
SELECT 10 * FLOOR( score / 10  ) AS  Bottom,
```

Read the entire item

Last updated 20 Oct 2014

Back to the top | Browse the book | Buy the book | Feedback

History of world records

In a table of track event results, rows marked as 'WR' in the `Note` column represent world record times. How would we retrieve the history of those world records?

```
drop table if exists results;
CREATE TABLE results (
  ID int UNSIGNED PRIMARY KEY AUTO_INCREMENT,
  Name varchar(50) COLLATE utf8_unicode_ci NOT NULL,
  Date date NOT NULL,
  Time int NOT NULL,
  Note varchar(50) COLLATE utf8_unicode_ci NOT NULL
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci AUTO_INCREMENT=14 ;

INSERT INTO results (ID, Name, Date, Time, Note) VALUES
(1, 'Bill', '2012-01-01', 58, 'WR'),(2, 'John', '2012-01-01', 59, ''),
```

Read the entire item

Last updated 23 Oct 2012

Back to the top *Browse the book* *Buy the book* *Feedback*

Pivot table basics: rows to columns

From table `tbl( class, member )`, you want to cross-tabulate all classes with their members. In SQL terms, you *aggregate members over classes*. In MySQL:

```
SELECT class,GROUP_CONCAT(member)
FROM tbl
GROUP BY class;
```

With that simple query, you're halfway toward cross-tabulation, halfway to implementing a simple `CUBE`, and halfway to basic entity-attribute-value (EAV) logic. This is easier to see if we have two columns, rather than just one, to tabulate against the grouping column:

```
DROP TABLE IF EXISTS tbl;
```

Read the entire item

Last updated 16 Feb 2011

Back to the top *Browse the book* *Buy the book* *Feedback*

Automate pivot table queries

You have a `sales` table listing product, salesperson and amount:

```
DROP TABLE IF EXISTS sales;
CREATE TABLE sales (
  id int(11) default NULL,
  product char(5) default NULL,
  salesperson char(5) default NULL,
  amount decimal(10,2) default NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
INSERT INTO sales VALUES
  (1,'radio','bob','100.00'),
  (2,'radio','sam','100.00'),
  (3,'radio','sam','100.00'),
```

Read the entire item

Last updated 17 Aug 2015

Back to the top *Browse the book* *Buy the book* *Feedback*

Group column statistics in rows

A *pivot* (or *crosstab*, or *contingency*) table aggregates sets of column values into rows of statistics, and *pivots* target value statistics on partitioning criteria defined

by any available data.

Spreadsheet applications have intuitive point-and-click interfaces for generating pivot tables. RDBMSs generally do not. The task looks difficult in SQL, though, only until you have coded a few.

If you ported the Microsoft sample database `Northwind` to your MySQL database (as described in chapter 11 of Get It Done with MySQL), you can execute this example step by step. Even if you haven't ported Northwind, the example is easy to follow.

Amongst the tables in the Northwind database are:

```
employees(employeeID, lastname, firstname, ...)
orders(orderID, customerId, employeeID, orderdate, ...)
```

Read the entire item

Last updated 15 Apr 2011

Back to the top *Browse the book* *Buy the book* *Feedback*

Monthly expenses

You have four tables to track revenue and expenses—bankaccount, cash, accountitem, accountcategory:

```
drop table if exists accountitem,accountcategory,bankaccount,cash;
create table accountitem(
  itemid int primary key auto_increment,itemname char(32),itemcatid int
);
create table accountcategory(
  categoryid int primary key auto_increment,categoryname char(32),isexpense bool
);
create table bankaccount(
  id int auto_increment primary key,amount decimal(12,2),itemid int,entrydate date
);
create table cash(
```

Read the entire item

Last updated 16 Feb 2010

Back to the top *Browse the book* *Buy the book* *Feedback*

Monthly sales

The pivot logic that reports expenses by month and year works just as well for sales. In the usual order entry system with customers, orders, products, orders, order items and payments (leaving out irrelevant details) ...

```
drop table if exists items, payments, products, orders, customers;
create table customers(cust_id int primary key) engine=innodb;
create table orders(
  order_id int primary key,
  cust_id int,
  order_date date,
  foreign key(cust_id) references customers(cust_id)  -- ORDER->CUST FK
) engine=innodb;
create table products(prod_id int primary key) engine=innodb;
create table items(
  item_id int primary key,
```

Read the entire item

Last updated 29 Feb 2012

Back to the top *Browse the book* *Buy the book* *Feedback*

Pivot table using math tricks

http://en.wikibooks.org/wiki/Programming:MySQL/Pivot_table

Last updated 22 May 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

Pivot table with CONCAT

Here is a MySQL pivot table query for room bookings by weekday:

```
SELECT slot
  , max(if(day=1, concat(subject,' ',room), '')) as day1
  , max(if(day=2, concat(subject,' ',room), '')) as day2
  , max(if(day=3, concat(subject,' ',room), '')) as day3
  , max(if(day=4, concat(subject,' ',room), '')) as day4
  , max(if(day=5, concat(subject,' ',room), '')) as day5
from schedule
group by slot
```

Read the entire item

Last updated 22 May 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

Pivot table without GROUP_CONCAT

Data designs often require flexibility in numbers and names of data points per instance row: instead of saving all the data points belonging to a key value in a single row, you save each data point as a name-value pair in its own row.

Thus given table user_class(user_id INT, class_id CHAR(20), class_value CHAR(20)) with these rows:

```
user_id  class_id   class_value
1        firstname  Rogier
1        lastname   Marat
2        firstname  Jean
2        lastname   Smith
```

Read the entire item

Last updated 22 May 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

Backslashes in data

Backslashes multiply weirdly:

```
SELECT 'a\b' RLIKE 'a\b';
```

returns 1, as does...

```
SELECT 'a\\b' RLIKE 'a\\\\b';
```

because in a pair of backslashes, the second is not escaped by the first, so to compare two literals you double each backslash in the `RLIKE` argument. But if you are querying a table for such a string from the MySQL client, this doubling happens twice--once in the client, and once in the database--so to find a *column* value matching `'a\\b'`, you need to write...

```
SELECT desc FROM xxx WHERE desc RLIKE 'aa\\\\\\\\bb';
```

That's *eight* backslashes to match two!

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Compare data in two tables

This query `UNIONS` queries for matching column names from two tables, and keeps just those rows which occur once in the union. Those are the rows with unmatched data. Customise your column list { id, col1, col2, col3 ...} as desired, but usually you'll want it to start with the primary key:

```
SELECT MIN(TableName) as TableName, id, col1, col2, col3, ...
FROM (
  SELECT 'Table a' as TableName, a.id, a.col1, a.col2, a.col3, ...
  FROM a
  UNION ALL
  SELECT 'Table b' as TableName, b.id, b.col1, b.col2, b.col3, ...
  FROM b
) AS tmp
GROUP BY id, col1, col2, col3, ...
HAVING COUNT(*) = 1
ORDER BY 1;
```

Read the entire item

Last updated 26 Nov 2010

Back to the top | Browse the book | Buy the book | Feedback

Show rows where column value changed

SQL is set-oriented, but it can solve row-by-row problems. Suppose you need to retrieve only the rows that differ from immediately previous rows given some ordering spec:

```
drop table if exists t;
create table t (
  p char(3),
  d date
);
insert into t values
('50%','2008-05-01'),
('30%','2008-05-02'),
('30%','2008-05-03'),
('50%','2008-05-04'),
('50%','2008-05-05'),
```

Read the entire item

Last updated 11 Aug 2010

Back to the top | Browse the book | Buy the book | Feedback

Using bit operators

If you store `ipaddress` values in a 32-bit unsigned int column, you can retrieve octets with bit-shifting:

```
select
  ipAddress,
  (ipAddress >> 24) as firstOctet,
  (ipAddress>>16) & 255 as secondOctet,
  (ipAddress>>8) & 255 as thirdOctet,
  ipAddress & 255 as fourthOctet
from ...
```

Last updated 17 Feb 2011

Back to the top | Browse the book | Buy the book | Feedback

Compare structures of two tables

To compare columns by name and ordinal position in tables test.t1 and test.t2:

```
SELECT
  MIN(TableName) AS 'Table',
  column_name AS 'Column',
  ordinal_position AS 'Position'
FROM (
  SELECT
    't1' as TableName,
    column_name,
    ordinal_position
  FROM information_schema.columns AS i1
  WHERE table_schema='test' AND table_name='t1'
```

Read the entire item

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Compare two databases

One of EF Codd's rules for relational databases is the no-back-door rule: all info about tables should be accessible only by a query on tables. Since version 5, the MySQL implementation of `information_schema` (I_S) helps meet Codd's requirement. I_S supplies metadata in tables, so it's the first place to look for how to compare the structures of two databases.

Elsewhere on this page there is a simple query template for comparing data in two structurally similar tables:

```
SELECT MIN(TableName) as TableName, id, col1, col2, col3, ...
FROM (
  SELECT 'Table a' as TableName, a.id, a.col1, a.col2, a.col3, ...
  FROM a
  UNION ALL
  SELECT 'Table b' as TableName, b.id, b.col1, b.col2, b.col3, ...
  FROM b
) AS tmp
```

Read the entire item

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Database size

```
SELECT
  table_schema AS 'Db Name',
  Round( Sum( data_length + index_length ) / 1024 / 1024, 3 ) AS 'Db Size (MB)',
  Round( Sum( data_free ) / 1024 / 1024, 3 ) AS 'Free Space (MB)'
FROM information_schema.tables
GROUP BY table_schema ;
```

Last updated 18 Jun 2010

Back to the top | Browse the book | Buy the book | Feedback

Find the size of all databases on the server

This is based on a query Mark Leith posted to the MySQL General Discussion list.

```
DROP VIEW IF EXISTS dbsize;
CREATE VIEW dbsize AS
SELECT
  s.schema_name AS 'Schema',
  SUM(t.data_length) AS Data,
  SUM( t.index_length ) AS Indexes,
  SUM(t.data_length) + SUM(t.index_length) AS 'Mb Used',
  IF(SUM(t.data_free)=0,'',SUM(t.data_free)) As 'Mb Free',
```

```
  IF( SUM(t.data_free)=0,
      '',
      100 * (SUM(t.data_length) + SUM(t.index_length)) / ((SUM(t.data_length)+SUM(t.index_length) + SUM(IFNULL(t.data_free,0))) )
    ) AS 'Pct Used',
  COUNT(table_name) AS Tables
FROM information_schema.schemata s
LEFT JOIN information_schema.tables t ON s.schema_name = t.table_schema
GROUP BY s.schema_name
WITH ROLLUP
```

Last updated 22 May 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

List databases, tables, columns

```
SELECT
  t.table_schema AS 'Database'
  ,t.table_name AS 'Table'
  ,t.table_type AS 'Table Type'
  ,c.column_name AS 'Column'
  ,c.data_type AS 'Data Type'
FROM information_schema.tables t
JOIN information_schema.columns c ON t.table_schema = c.table_schema AND t.table_name = c.table_name
WHERE t.table_schema NOT IN( 'mysql','information_schema')
ORDER BY t.table_schema,t.table_type,t.table_name,c.ordinal_position;
```

Last updated 22 May 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

List differences between two databases

Given two databases named @db1 and @db2:

```
SELECT
  MIN(table_name) as TableName,
  table_catalog,table_schema,table_name,column_name,
  ordinal_position,column_default,is_nullable,
  data_type,character_maximum_length,character_octet_length,
  numeric_precision,numeric_scale,character_set_name,
  collation_name,column_type,column_key,
  extra,privileges,column_comment
FROM (
  SELECT 'Table a' as TableName,
  table_catalog,table_schema,table_name,column_name,
```

Read the entire item

Last updated 22 May 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

List users of a database

```
DROP PROCEDURE IF EXISTS ListDbUsers;
DELIMITER |
CREATE PROCEDURE ListDbUsers( dbname CHAR(64) )
  SELECT host,user
  FROM mysql.user
  WHERE Select_priv = 'Y'
       OR Insert_priv = 'Y'
     OR Update_priv = 'Y'
     OR Delete_priv = 'Y'
     OR Create_priv = 'Y'
     OR Drop_priv = 'Y'
     OR Reload_priv = 'Y'
     OR Shutdown_priv = 'Y'
```

Read the entire item

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Rename Database

Sometimes it's necessary to rename a database. There are two crucial reasons there's no such command in MySQL: there's no way to lock a whole database, and `information_schema` isn't transactional.

MySQL 5.1.7 introduced a `RENAME DATABASE` command, but the command left several unchanged database objects behind, and was found to lose data, so it was dropped in 5.1.23.

The operating system's view of a database is that it's just another folder on the disk. That may tempt you to think you could just rename it. *Don't*.

Renaming a database is perfectly safe if and only if:

- it has no tables with foreign keys that reference, or are referenced by, a table in another database;
- it has no procedures, functions or triggers that reference objects in another database;
- no other database has tables, functions, procedures or triggers that reference this database.

So there is one safe way to rename a database: dump it completely ...

Read the entire item

Last updated 29 Nov 2011

Back to the top | Browse the book | Buy the book | Feedback

Replace text in all string columns of a database

Generate SQL to replace 'old_text' with 'new_text' in all character columns in 'dbname':

```
Select Concat( 'update ', table_schema, '.', table_name,
               ' set ', column_name,
               '=replace(', column_name, ','''old_text''','''new_text''');'
             )
From information_schema.columns
Where (data_type Like '%char%' or data_type like '%text' or data_type like '%binary')
  And table_schema = 'dbname';
```

Last updated 23 May 2010

Back to the top | Browse the book | Buy the book | Feedback

Foreign key basics

Referential integrity (RI) is data consistency between related tables. `ACID`-compliant database engines like InnoDB provide foreign keys to automate RI maintenance. Simple example: `customers` have `orders`, which have `items`; by defining a `customers` foreign key in the `orders` table and an `orders` foreign key in the `items` table, you ensure that no `orders` or `items` row gets orphaned.

SQL implements FKs as key objects defined by `CREATE TABLE` and `ALTER TABLE`. MySQL syntax is:

```
[CONSTRAINT symbol] FOREIGN KEY [constraintID] (keycolumnname, ...)
REFERENCES tbl_name (keycolumnname, ...)
[ ON DELETE { RESTRICT | CASCADE | SET NULL | NO ACTION } ]
[ ON UPDATE { RESTRICT | CASCADE | SET NULL | NO ACTION } ]
```

To drop a FK in MySQL you need to know the `CONSTRAINT symbol`, so it's good

practice to adopt a standard naming convention and use it here.

Read the entire item

Last updated 09 Oct 2013

Back to the top | Browse the book | Buy the book | Feedback

Change or drop a foreign key

To change a foreign key, first drop it, then declare the new, revised foreign key. The syntax for declaring a foreign key is ...

```
[CONSTRAINT [constraint_name]]
FOREIGN KEY [key_name] (keycol_name,...) reference_definition
```

and the syntax for dropping one is ...

```
DROP FOREIGN KEY constraint_name
```

Notice that you can omit the `CONSTRAINT` when you declare a foreign key, but the *only* way to `DROP` a foreign key is to reference it by the `constraint_name` which you probably never specified!

There should be a circle of hell reserved for designers who build inconsistencies like this into their tools. The only way round this one is to run `SHOW CREATE TABLE` to find out what the foreign key's `constraint_name` is, so you can write the `DROP` statement. Here is a wee test case:

```
drop table if exists a,b;
create table a(i int primary key)engine=innodb;
create table b(i int,foreign key(i) references a(i)) engine=innodb;
show create table\G

CREATE TABLE `b` (
  `i` int(11) DEFAULT NULL,
  KEY `i` (`i`),
  CONSTRAINT `b_ibfk_1` FOREIGN KEY (`i`) REFERENCES `a` (`i`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1

-- drop and recreate the FK:
alter table b drop foreign key b_ibfk_1;
alter table b add foreign key(i) references a(i) on update cascade;
show create table b\G

Create Table: CREATE TABLE `b` (
  `i` int(11) DEFAULT NULL,
  KEY `i` (`i`),
  CONSTRAINT `b_ibfk_1` FOREIGN KEY (`i`) REFERENCES `a` (`i`) ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1

drop table a,b;
```

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Find child tables

Starting with MySQL 5, you can find all tables which have foreign key references to a given table with an `information_schema` query, here encapsulated in a stored procedure which takes two arguments, a database name and a table name. When the table argument is blank or `NULL`, the procedure returns all parent-child links where the parent table is in the specified database; otherwise it returns parent-child links for the specified parent table:

```
DROP PROCEDURE IF EXISTS ListChildren;
DELIMITER |
CREATE PROCEDURE ListChildren( pdb CHAR(64), ptbl CHAR(64) )
BEGIN
  IF ptbl = '' OR ptbl IS NULL THEN
    SELECT
      c.table_schema as 'Parent Schema',
```

```
      u.referenced_table_name as 'Parent Table',
      u.referenced_column_name as 'Parent Column',
      u.table_schema as 'Child Schema',
      u.table_name as 'Child Table',
```

Read the entire item

Last updated 28 Jan 2011

Back to the top | Browse the book | Buy the book | Feedback

Find cross-database foreign keys

```
SELECT
  u.table_schema as 'RefSchema',
  u.table_name as 'RefTable',
  u.column_name as 'RefColumn',
  u.referenced_table_schema as 'Schema',
  u.referenced_table_name as 'Table',
  u.referenced_column_name as 'Column'
FROM information_schema.table_constraints AS c
JOIN information_schema.key_column_usage AS u
USING( constraint_schema, constraint_name )
WHERE c.constraint_type='FOREIGN KEY' AND u.table_schema <> u.referenced_table_schema;
```

To find them for a particular database, add the `WHERE` condition:

```
AND 'dbname' IN(u.table_schema, u.referenced_table_schema)
```

Last updated 27 Nov 2011

Back to the top | Browse the book | Buy the book | Feedback

Find parent tables

List tables which are referenced by foreign key constraints in a given table.This is a simple query on two information_schema tables: table_constraints and key_column_usage. It is easy to parameterise, so we show it in stored procedures. The first sproc lists all foreign key references in a database. The second lists all foreign key references for a table.

```
CREATE PROCEDURE ListParentsForDb( pdb CHAR(64) )
  SELECT
   u.table_schema AS 'Schema',
   u.table_name AS 'Table',
   u.column_name AS 'Key',
   u.referenced_table_schema AS 'Parent Schema',
   u.referenced_table_name AS 'Parent table',
   u.referenced_column_name AS 'Parent key'
  FROM information_schema.table_constraints AS c
  JOIN information_schema.key_column_usage AS u
  USING( constraint_schema, constraint_name )
```

Read the entire item

Last updated 16 Mar 2013

Back to the top | Browse the book | Buy the book | Feedback

Add auto-incrementing primary key to a table

The steps are: (i) recreate the table, populating a new column from an incrementing user variable, then (ii) alter the table to add auto_increment and primary key properties to the new column. So given table t with columns named `dt` and `observed`...

```
DROP TABLE IF EXISTS t2;
SET @id=0;
CREATE TABLE t2
  SELECT @id:=@id+1 AS id, dt, observed FROM t ORDER BY dt;
```

```
ALTER TABLE t2
  MODIFY id INT AUTO_INCREMENT PRIMARY KEY;
DROP TABLE t;
RENAME TABLE t2 TO t;
```

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Auto-increment: reset next value

ALTER TABLE tbl SET AUTO_INCREMENT=val;

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Find primary key of a table

To retrieve primary keys of db.tbl...

```
SELECT k.column_name
FROM information_schema.table_constraints t
JOIN information_schema.key_column_usage k
USING (constraint_name,table_schema,table_name)
WHERE t.constraint_type='PRIMARY KEY'
  AND t.table_schema='db'
  AND t.table_name='tbl'
```

For pre-5 versions of MySQL:

```
SHOW INDEX FROM tbl
WHERE key_name='primary';
```

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Show Create Trigger

MySQL added a Show Create Trigger command in 5.1.21. If you use an earlier MySQL version, here is a stored procedure which behaves like Show Create Trigger:

```
DROP PROCEDURE IF EXISTS ShowCreateTrigger;
DELIMITER go
CREATE PROCEDURE ShowCreateTrigger( IN db CHAR(64), IN tbl CHAR(64) )
  BEGIN
    SELECT
      CONCAT(
        'CREATE TRIGGER ',trigger_name, CHAR(10),
        action_timing,' ', event_manipulation, CHAR(10),
        'ON ',event_object_schema,'.',event_object_table, CHAR(10),
        'FOR EACH ROW', CHAR(10),
        action_statement, CHAR(10)
```

Read the entire item

Last updated 08 Jun 2015

Back to the top | Browse the book | Buy the book | Feedback

Show Table Status equivalent from information_schema

Fill in schema and table names in ...

```
SELECT
  table_name,
  engine,
```

```
  version,
  row_format,
  table_rows,
  avg_row_length,
  data_length,
  max_data_length,
  index_length,
  data_free,
  auto_increment,
  create_time,
  update_time,
  check_time,
  table_collation,
  checksum,
  create_options,
  table_comment
FROM information_schema.tables
where table_schema='???' AND table_name='???';
```

Last updated 09 Oct 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

Show Tables

The MySQL `SHOW TABLES` command is fine, but sometimes we want a little more information.

This simple stored procedure lists the table name, engine type, version, collation and rowcount for every table in a database. (Individual databases come and go, so we keep all such database-wide stored routines in a `system` database.)

```
DROP PROCEDURE IF EXISTS showtables;
CREATE PROCEDURE showtables()
  SELECT
    table_name AS 'Table',
    IFNULL(engine, 'VIEW') AS Engine,
    version AS Version,
    table_collation AS Collation,
    table_rows AS Rows
```

Read the entire item

Last updated 22 Nov 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

Age in years

Given a birthdate in `@dob`, here are two simple formulae for age in years:

```
Date_format( From_Days( To_Days(Curdate()) - To_Days(@dob) ), '%Y' ) + 0
```

```
Year(Curdate()) - Year(@dob) - ( Right(Curdate(),5) < Right(@dob,5) )
```

and here is one for age in years to two decimal places, ignoring day of month:

```
Round((((Year(now()) - Year(@dob)))*12 + (((Month(now()) - Month(@dob)))))/12, 2)
```

Last updated 27 Dec 2012

Back to the top *Browse the book* *Buy the book* *Feedback*

Compute date from year, week number and weekday

```
SET @yr=2012, @wk=26, @day=0;
SELECT Str_To_Date( Concat(@yr,'-',@wk,'-',If(@day=7,0,@day) ), '%Y-%U-
%w' ) AS Date;
+------------+
| Date       |
+------------+
```

```
| 2012-06-24 |
+------------+
```

Last updated 01 Jul 2012

Back to the top *Browse the book* *Buy the book* *Feedback*

Count business days between two dates

The simplest support for counting business days between any two dates is a calendar table `calendar(d date,isholiday bool)` populated for all days in all possibly relevant years. Then the following query gives the inclusive number of business days between dates `dStart` and `dStop`:

```
SELECT COUNT(*)
FROM calendar
WHERE d BETWEEN dStart AND dStop
  AND DAYOFWEEK(d) NOT IN(1,7)
  AND isholiday=0;
```

If that solution is not available, you have to do with a weekday count, which this function (corrected 6 Jul 2009) computes:

Read the entire item

Last updated 10 Jul 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

Count Tuesdays between two dates

Date arithmetic is deceptively hard. One way to appreciate the difficulties is to read Chapter 21 in our book. Another is to try to calculate the number of Tuesdays (or another weekday) between two dates. It's not a back-of-the-napkin problem.

An earlier formula we had for this problem sometimes gave incorrect results. As a debugging aid, we wrote a brute force calculator for the problem:

```
SET GLOBAL log_bin_trust_function_creators=1;
DROP FUNCTION IF EXISTS DayCount;
DELIMITER |

CREATE FUNCTION DayCount( d1 DATE, d2 DATE, daynum SMALLINT ) RETURNS INT
BEGIN
  DECLARE days INT DEFAULT 0;
  IF D1 IS NOT NULL AND D2 IS NOT NULL THEN
```

Read the entire item

Last updated 22 May 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

Date of first day of next month

Date of first day of next month:

```
concat(left(curdate() + interval 1 month, 8), '-01');
```

Date of first day of previous month:

```
concat(left(curdate() - interval 1 month, 8), '-01');
```

Last updated 29 Mar 2012

Back to the top *Browse the book* *Buy the book* *Feedback*

Date of first Friday of next month

Assuming a calendar table `calendar(date DATE)` with one row per date through the relevant period...

```
SET @d = NOW();
SELECT MIN(date) AS 1stFridayOfMonth
FROM calendar
WHERE YEAR(date) = IF( MONTH(@d) = 12, 1+YEAR(@d), YEAR(@d) )
  AND MONTH(date) = IF( MONTH(@d) = 12, 1, MONTH(@d) + 1 )
  AND WEEKDAY(date)=4;
```

Last updated 22 May 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

Date of Monday in a given week of the year

Given a valid year value `@y` and a week value `@w` between 1 and 53, this formula gives the date of Monday in week `@w` of year `@y`:

```
adddate( makedate(@y,@w*7), interval 2-dayofweek(makedate(@y,7)) day )
```

Last updated 29 Nov 2011

Back to the top *Browse the book* *Buy the book* *Feedback*

Date of Monday of this week

```
set @d = '2011-11-29';
select adddate( date(@d), interval 2-dayofweek(@d) day ) as 1stdayofweek;
+---------------+
| 1stydayofweek |
+---------------+
| 2011-11-28    |
+---------------+
```

Last updated 29 Nov 2011

Back to the top *Browse the book* *Buy the book* *Feedback*

Datetime difference

Find the difference between two datetime values in seconds, minutes, hours or days. If dt1 and dt2 are datetime values of the form 'yyyy-mm-dd hh:mm:ss', the number of seconds between dt1 and dt2 is

```
UNIX_TIMESTAMP( dt2 ) - UNIX_TIMESTAMP( dt1 )
```

To get the number of minutes divide by 60, for the number of hours divide by 3600, and for the number of days, divide by 3600 * 24.

Last updated 22 May 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

Duration in years, months, days and time

```
DROP FUNCTION IF EXISTS PeriodLen;
DROP FUNCTION IF EXISTS NumLabel;
DELIMITER go
CREATE FUNCTION PeriodLen( dt1 datetime, dt2 datetime ) RETURNS CHAR(128)
BEGIN
  DECLARE yy,m0,mm,d0,dd,hh,mi,ss,t1 BIGINT;
  DECLARE dtmp DATETIME;
  DECLARE t0 TIMESTAMP;
  SET yy = TIMESTAMPDIFF(YEAR,dt1,dt2);
  SET m0 = TIMESTAMPDIFF(MONTH,dt1,dt2);
```

```
  SET mm = m0 MOD 12;
  SET dtmp = ADDDATE(dt1, interval m0 MONTH);
  SET d0 = TIMESTAMPDIFF(DAY,dt1,dt2);
```

Read the entire item

Last updated 20 Jun 2012

Back to the top | Browse the book | Buy the book | Feedback

Julian date

Unix_Timestamp(datetimevalue) / (60*60*24)) + 2440587.5

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Last business day before a reference date

Given a date value in *datetimecol* ...

```
SELECT
  @refday := datetimecol,
  @dow := DAYOFWEEK(@refday) AS DOW,
  @subtract := IF( @dow = 1, 2, IF( @dow = 2, 3, 1 )) AS MINUS,
  @refday - INTERVAL @subtract DAY AS LastBizDay
FROM ... etc
```

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Make a calendar table

You need a calendar table for joins to datetime data in other tables:

```
create table calendar ( dt datetime primary key );
```

An elegant method of generating any desired number of sequential values, posted by Giuseppe Maxia on his blog, is ...

- Create three dummy rows in a View.
- Cross join them to make 10 dummy rows.
- Cross join those to make 100, 1,000 or however many you need.

So to give the calendar table a million rows at one-hour intervals starting on 1 Jan 1970:

Read the entire item

Last updated 13 Jun 2013

Back to the top | Browse the book | Buy the book | Feedback

Scope to the week of a given date

To scope a query to the calendar week (Sunday through Saturday) of a date value `@d`, write ...

```
... WHERE d BETWEEN AddDate(@d,-DayOfWeek(@d)+1) and AddDate(@d,7-
DayOfWeek(@d)) ...
```

Last updated 29 Dec 2009

Back to the top | Browse the book | Buy the book | Feedback

Sum accumulated time by date

You track resource booking periods. You need a query to report daily usage for a given resource.

First the problem of calculating per-diem usage. Call the starting datetime of a booked period `pStart`, and its ending datetime `pEnd`. Then for a given date `pDate`, if the period began before `pDate`. then `pDate` usage begins at 00:00:00, otherwise it starts at `pStart`; likewise if the period extends past `pDate`, then `pDate` usage ends at midnight on `pDate`, otherwise it ends at `pEnd`. Therefore the period begins at...

```
IF( pStart < pDate, CAST(pDate AS DATETIME ), pStart )
```

and ends at...

Read the entire item

Last updated 20 Oct 2014

Back to the top | Browse the book | Buy the book | Feedback

Sum booking days for a given month

A table tracks beginning and ending booking dates `dfrom` and `dto`. How to sum booking durations in days for a given month, say July?

A booking may begin and end in the target month, or begin, or end, or neither:

```
select
  sum(
    case
       when month(dfrom)=7 and month(dfrom)=month(dto) then datediff(dto,dfrom)
       when month(dfrom)=7                             then datediff(last_day(dfrom),dfrom)
       when month(dto)=7                               then datediff(dto,date_format(dto,'%Y-
%m-01'))
       else 0
     end
  ) as DaysInJuly
  from ...
```

Last updated 10 Nov 2014

Back to the top | Browse the book | Buy the book | Feedback

Sum time values

```
SELECT SEC_TO_TIME( SUM( TIME_TO_SEC( time_col ))) AS total_time
FROM tbl;
```

Summing values like '12:65:23' produces meaningless results.

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

The date of next Thursday

Given a date and its weekday number (1=Sunday, ..., 7=Saturday), there are three possibilities:
1. Today is Thursday: then next Thursday is 7 days from now.
2. Today is before Thursday: then next Thursday is (5 minus today's weekday number) from now.
3. Today is after Thursday: then next Thursday is 7 + (5 minus today's weekday number).

```
set @d=curdate();
set @n = dayofweek(curdate());
select
```

```
  @d:=adddate(curdate(),0) as date,
  @n:=dayofweek(adddate(curdate(),0)) as weekday,
```

Read the entire item

Last updated 02 Dec 2009

Back to the top | Browse the book | Buy the book | Feedback

Track when a value changed

You have a table that tracks a value and the time when the value was measured ...

```
drop table if exists changes;
create table changes(time time,value int);
insert into changes values
('00:00', 0 ),
('01:05', 1 ),
('01:09', 1 ),
('01:45', 1 ),
('02:24', 0 ),
('12:20', 1 ),
('12:40', 0 ),
('14:32', 0 ),
```

Read the entire item

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

What month does a week fall in?

"The" month of a week is ambiguous if the week straddles months. If we adopt the convention that the month of a week is the month of its beginning Sunday, then on 29 Nov 2009 ...

```
SET @weekno = Month CurDate() );
SET @date = AddDate('2009-01-01', 7*@weekno );
SET @day = DayOfWeek( @date );
SET @datecomp = IF( @day = 1, @date, AddDate( @date, 1-@day ));
SELECT @date,@day,@datecomp,Month(@datecomp) AS month;
+------------+------+------------+-------+
| @date      | @day | @datecomp  | month |
+------------+------+------------+-------+
| 2009-12-03 |    5 | 2009-11-29 |    11 |
+------------+------+------------+-------+
```

Read the entire item

Last updated 30 Nov 2009

Back to the top | Browse the book | Buy the book | Feedback

YearMonth()

We often need to compute datetimes based on year and month. This tiny function simplifies such queries:

```
set global log_bin_trust_function_creators=1;
create function yearmonth(d date) returns int
return 100*year(d)+month(d);
```

Then to find date values within the three-month period bounded by the first day of last month and the last day of next month, write ...

```
select d
from tbl
where yearmonth(d) between yearmonth(curdate()-
interval 1 month) and yearmonth(curdate()+interval 1 month);
```

Last updated 23 Aug 2010

Back to the top | Browse the book | Buy the book | Feedback

Audit trails and point-in-time architecture

http://www.artfulsoftware.com/infotree/Transaction time validity in MySQL.pdf

Last updated 31 Jul 2012

Back to the top | Browse the book | Buy the book | Feedback

Find overlapping periods

You have a table of visits, and you would like to display the time periods during which there are visit time overlaps.

```
drop table if exists visits;
create table visits(id int primary key,start datetime,end datetime);
insert into visits values
(1, '2008-09-01 15:01', '2008-09-01 15:04'),
(2, '2008-09-01 15:02', '2008-09-01 15:09'),
(3, '2008-09-01 15:12', '2008-09-01 15:15'),
(4, '2008-09-01 16:11', '2008-09-01 16:23'),
(5, '2008-09-01 16:19', '2008-09-01 16:25'),
(6, '2008-09-01 17:52', '2008-09-01 17:59'),
(7, '2008-09-01 18:18', '2008-09-01 18:22'),
(8, '2008-09-01 16:20', '2008-09-01 16:22'),
```

Read the entire item

Last updated 05 Sep 2009

Back to the top | Browse the book | Buy the book | Feedback

Find sequenced duplicates

A table that tracks time periods may require period uniqueness. That means it has no sequenced duplicates.

If a table has columns processID, start_date and end_date, those three columns are period unique if there exists no pair of rows with the same processID and overlapping start_date and end_date values. If there is such a pair of rows, the table exhibits *sequenced duplication*.

Another way of saying it: if an instant is the smallest datetime unit of start_date and end_date columns, then if there are no sequenced duplicates, there is exactly one processID value at any instant.

Here is a query to find sequenced duplicates for those columns:

```
SELECT t.processid
FROM tbl t
WHERE EXISTS (
  SELECT * FROM tbl AS t3
  WHERE t3.processid IS NULL
)
OR EXISTS (
  SELECT * FROM tbl AS t1
  WHERE 1 < (
    SELECT COUNT(processid)
    FROM tbl AS t2
    WHERE t1.processid = t2.processid
      AND t1.start_date < t2.end_date
      AND t2.start_date < t1.end_date
  )
);
```

Last updated 26 Dec 2012

Back to the top | Browse the book | Buy the book | Feedback

In or out at a given date and time?

Employees punch in and out. You track these events with ...

```
drop table if exists tbl;
create table tbl( empno smallint, clockdate date, clocktime time, clocktype char(1) );
insert into tbl values
(1,  '2014-05-15', '09:00:00', 'I' ),
(1,  '2014-05-15', '11:00:00', 'O' ),
(1,  '2014-05-15', '12:30:00', 'I' ),
(1,  '2014-05-15', '19:00:00', 'O' );
```

Was employee 1 in or out at 12:30pm on 15 May 2014? Use a self-join to play the in and out events against the given datetime ...

```
select if(count(1),'Yes','No' ) as InStore
from tbl a
join tbl b using(empno, clockdate)
where empno=1
  and a.clockdate='2014-5-15'
  and a.clocktime<='12:30:00' and a.clocktype='I'
  and b.clocktime>'12:30:00' and b.clocktype='O';
+---------+
| InStore |
+---------+
| Yes     |
+---------+
```

Last updated 10 Jun 2014

Back to the top | Browse the book | Buy the book | Feedback

Peak visit counts by datetime period

You have a `visits` table (`id int, start datetime, end datetime`), and you wish to track peak visit counts. A simple solution is to self-join on non-matching IDs and overlapping visit times, group by ID, then order by the resulting counts:

```
SELECT a.id,group_concat(b.id) as Overlaps, count(b.id)+1 as OverlapCount
FROM visits a
JOIN visits b on a.id < b.id and a.start < b.end and b.start < a.end
GROUP BY a.id
ORDER BY OverlapCount DESC;
```

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Sum for time periods

A table tracks attendance at some location:

```
drop table if exists t;
create table t(interval_id int,start datetime,end datetime, att int);
insert into t values
(1,'2007-01-01 08:00:00','2007-01-01 12:00:00',5 ),
(2,'2007-01-01 13:00:00','2007-01-01 17:00:00',10),
(3,'2007-01-01 10:00:00','2007-01-01 15:00:00',15),
(4,'2007-01-01 14:00:00','2007-03-07 19:00:00',20);
select * from t;
+-------------+---------------------+---------------------+------+
| interval_id | start               | end                 | att  |
+-------------+---------------------+---------------------+------+
```

Read the entire item

Last updated 03 Mar 2015

Back to the top | Browse the book | Buy the book | Feedback

Appointments available

Given a clinic of physicians, patients and appointments, how to find an available appointment time for a given physician?

This is a variant of the [Not] Exists query pattern. Though we can write it with subqueries, performance will be crisper with a join. But finding data that is not there requires a join to data which *is* there. So in addition to tables for appointments, doctors and patients, we need a table of all possible appointment datetimes. Here's a schema illustrating the idea ...

```
CREATE TABLE a_dt (         -- POSSIBLE APPOINTMENT DATES AND TIMES
  d DATE,
  t TIME
);
CREATE TABLE a_drs (        -- DOCTORS
  did INT                   -- doctor id
);
CREATE TABLE a_pts (        -- PATIENTS
```

Read the entire item

Last updated 22 May 2009

Back to the top Browse the book Buy the book Feedback

Find available reservation periods

Given a bookings table where each row specifies one reservation period for one property, find the unbooked periods for a given property:

```
CREATE TABLE bookings( ID int, propertyID int, startDate date, endDate date );
INSERT INTO bookings VALUES
  (1,1,'2007-1-1','2007-1.15'),
  (2,1,'2007-1-20','2007-1.31'),
  (3,1,'2007-2-10','2007-2-17');
SELECT * FROM bookings;
+------+------------+------------+------------+
| ID   | propertyID | startDate  | endDate    |
+------+------------+------------+------------+
|    1 |          1 | 2007-01-01 | 2007-01-15 |
|    2 |          1 | 2007-01-20 | 2007-01-31 |
```

Read the entire item

Last updated 05 Mar 2015

Back to the top Browse the book Buy the book Feedback

Game schedule

List all possible home-away encounters of teams listed in a table.

```
SELECT t1.name AS Visiting,
       t2.name AS Home
FROM teams AS t1
STRAIGHT_JOIN teams AS t2
WHERE t1.ID <> t2.ID;
```

Last updated 22 May 2009

Back to the top Browse the book Buy the book Feedback

Is a given booking period available?

You rent vacation properties, tracking bookings with a table like this:

```
CREATE TABLE bookings( ID int, propertyID int, startDate date, endDate date );
INSERT INTO bookings VALUES (1,1,'2007-1-1','2007-1.15'),(2,1,'2007-1-20','2007-
1.31');
SELECT * FROM bookings;
```

```
+------+------------+------------+------------+
| ID   | propertyID | startDate  | endDate    |
+------+------------+------------+------------+
|    1 |          1 | 2007-01-01 | 2007-01-15 |
|    2 |          1 | 2007-01-20 | 2007-01-31 |
+------+------------+------------+------------+
```

Read the entire item

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Pivot table schedule

You have a schedule table (period, day, subject, room) with a primary key *period,day* to avoid duplicate bookings. You wish to display the schedule as periods, subjects and rooms in rows, and days of the week in columns.

```
SELECT
  period,
  MAX(IF(day=1, CONCAT(subject,' ',room), '')) AS Mon,
  MAX(IF(day=2, CONCAT(subject,' ',room), '')) AS Tue,
  MAX(IF(day=3, CONCAT(subject,' ',room), '')) AS Wed,
  MAX(IF(day=4, CONCAT(subject,' ',room), '')) AS Thu,
  MAX(IF(day=5, CONCAT(subject,' ',room), '')) AS Fri
FROM schedule
GROUP BY period
```

MAX() chooses existing over blank entries, and GROUP BY lines everything up on the same row.

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Display column values which occur N times

```
SELECT id
FROM tbl
GROUP BY id
HAVING COUNT(*) = N;
```

Change the HAVING condition to >1 to list duplicate values, etc.

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Display every Nth row

Display every Nth row in `tbl` where `id` is sequential in MySQL before version 4.1:

```
SELECT id
FROM tbl
GROUP BY id
HAVING MOD(id, N) = 0;
```

or

Read the entire item

Last updated 02 Sep 2015

Back to the top | Browse the book | Buy the book | Feedback

Trees, networks and parts explosions in MySQL

http://www.artfulsoftware.com/mysqlbook/sampler/mysqled1ch20.html

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Dijkstra's shortest path algorithm

Given a table of source-to-destination paths, each of whose nodes references a row in a nodes table, how do we find the shortest path from one node to another?

One answer is Dijkstra's algorithm (http://en.wikipedia.org/wiki/Dijkstra's_algorithm). Peter Larsson has posted a SQL Server implementation of it on the SQL Team Forum. Here is a MySQL implementation.

The DDL:

```
DROP TABLE IF EXISTS dijnodes,dijpaths;
CREATE TABLE dijnodes (
  nodeID int PRIMARY KEY AUTO_INCREMENT NOT NULL,
  nodename varchar (20) NOT NULL,
  cost int NULL,
```

Read the entire item

Last updated 26 Dec 2012

Back to the top | Browse the book | Buy the book | Feedback

Multiple trees in one table

Multiple trees can bury you in table glut. Can we combine them in one table and still query them conveniently? Yes. What's more, it's useful to combine edge list and nested sets representations of the same trees in one table. Here is a table definition for the job:

```
CREATE TABLE trees (
  root int(11) DEFAULT 0,         -- to which tree does this node belong?
  nodeID int(11) NOT NULL,        -- edge list representation of this node
  parentID int(11) DEFAULT NULL,
  level smallint(6) DEFAULT 0,    -- depth of this node in this tree
  lft int(11) DEFAULT 0,          -- nested sets representation of this node
  rgt int(11) DEFAULT 0,
  PRIMARY KEY (root,nodeID)
) ENGINE=InnoDB;
```

Read the entire item

Last updated 05 Sep 2011

Back to the top | Browse the book | Buy the book | Feedback

Tree query performance

http://www.artfulsoftware.com/infotree/treequeryperformance.pdf

Last updated 04 Apr 2010

Back to the top | Browse the book | Buy the book | Feedback

Trees of known depth

A tree is a hierarchy where each node except the root has one parent. A parent-child link is an *edge*. Edges in a tree of known depth can be queried with n-1 self-joins where n is the number of edges from top to bottom. Here is a simple example of depth 2:

```
drop table if exists t;
create table t(id int, parent int, ord int, title char(20));
```

```
insert into t values
(1, 0, 0, 'Root'),
(2, 1, 0, 'Home'),
(3, 1, 1, 'Projects'),
(5, 1, 2, 'Secret area'),
(4, 1, 3, 'Tutorials'),
(8, 1, 4, 'Example'),
(6, 4, 0, 'Computing'),
(7, 4, 1, 'Life');
```

Read the entire item

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Approximate joins

There are two main ways to reconcile payments against charges:

- *Open Item*: match payments against individual charges, typically by carrying the charge number in the payments table
- *Statement*: list and sum all charges and all payments, and show the difference as the outstanding balance.

The Open Item method needs a foolproof way to match payments to charges, but what if the customer neglected to return a copy of the invoice, or to write the invoice number on the cheque? Reconciliation staff spend much of their time resolving such problems.

Can we help? Yes! It won't be entirely foolproof, but it will drastically cut down the onerous work of reconciliation.

Here is DDL for a test case:

```
CREATE SCHEMA approx;
USE approx;
```

Read the entire item

Last updated 05 May 2010

Back to the top | Browse the book | Buy the book | Feedback

Cascading JOINs

Show parents, children and grandchildren including parents without children

```
SELECT parent.id AS ParentID,
       IFNULL(child.parent_id,') AS ChildParentID,
       IFNULL(child.id,') AS ChildID,
       IFNULL(grandchild.child_id,') AS GrandchildChildID
FROM parent
  LEFT JOIN child ON parent.id=child.parent_id
  LEFT JOIN grandchild ON child.id=grandchild.child_id;
```

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Classroom scheduling

You have *n* student classes of known size, and *m* classrooms of known size, where *m*>=*n*. What's the best algorithm for assigning as many classes as possible to rooms of adequate size?

It's a version of the combinatorial knapsack problem. It's known to be NP-complete, which means it's possible to verify any correct solution but there is no

known algorithm for quickly finding a correct solution. How then to proceed?

Early in 2010 Joe Celko resurrected the problem in a Simple Talk column, and challenged readers to improve on SQL Server solutions he'd published in the third edition of his "SQL for Smarties". Here's his small version of the problem modified for MySQL:

```
DROP TABLE IF EXISTS Rooms, Classes;
CREATE TABLE Rooms(
  room_nbr CHAR(2) NOT NULL PRIMARY KEY, room_size INTEGER NOT NULL
) ENGINE=MyISAM;
CREATE TABLE Classes(
```

Read the entire item

Last updated 28 Jan 2011

Back to the top Browse the book Buy the book Feedback

Data-driven joins

Data-driven table relationships are hard to maintain, but sometimes they cannot be avoided. How do we build joins for them? One way is to use a `CASE` statement in the `SELECT` list to handle the joining possibilities. In this example, the `parent.linktable` column determines the name of the table where a particular parent row's data is. The method is fine when the number of child tables is small:

```
USE test;
DROP TABLE IF EXISTS parent, child1, child2;

CREATE TABLE parent (
  id INT UNSIGNED PRIMARY KEY,
  linktable CHAR(64) NOT NULL
);
INSERT INTO parent VALUES (1, 'child1'), (2, 'child2');

CREATE TABLE child1 (
  id INT UNSIGNED PRIMARY KEY,
```

Read the entire item

Last updated 22 May 2009

Back to the top Browse the book Buy the book Feedback

Full Outer Join

A `FULL OUTER` join between tables a and b retrieves:

- all rows from a, with matching rows or nulls from b, and
- all rows from b, with matching rows or nulls from a

so for these tables:

```
DROP TABLE IF EXISTS a,b;
CREATE TABLE a(id int,name char(1));
CREATE TABLE b(id int,name char(1));
INSERT INTO a VALUES(1,'a'),(2,'b');
INSERT INTO b VALUES(2,'b'),(3,'c');
SELECT * FROM a;
+------+------+
| id   | name |
```

Read the entire item

Last updated 12 Oct 2014

Back to the top Browse the book Buy the book Feedback

Intersection and difference

MySQL implements `UNION`, but does not directly implement `INTERSECTION` or `DIFFERENCE`.

`INTERSECTION` is just an `INNER JOIN` on all columns:

```
drop table if exists a,b;
create table a(i int,j int);
create table b like a;
insert into a values(1,1),(2,2);
insert into b values(1,1),(3,3);
select * from a join b using(i,j);
+------+------+
| i    | j    |
```

Read the entire item

Last updated 30 Jun 2009

Back to the top | Browse the book | Buy the book | Feedback

Many-to-many joins

To model a many:many relationship between two tables a and b, you need a bridging table where each row represents one instance of an association between a row in a and a row in b, as in this example:

```
drop table if exists users,actions,useractions;
create table users(userid int primary key, username char(32));
insert into users values(1, 'James'),(2, 'Alex'),(3, 'Justin');
create table actions(actionid int primary key, action char(32));
insert into actions values(1, 'Login'),(2, 'Logout'),(3, 'Delete'),
(4, 'Promote');
create table useractions(uaid int primary key, userid int, actionid int);
insert into useractions values(1,1,1),(2,1,2),(3,3,4);

select u.username, a.action
from useractions ua
join users   u using (userid)
```

Read the entire item

Last updated 16 Mar 2010

Back to the top | Browse the book | Buy the book | Feedback

What else did buyers of X buy?

We often want to know how certain column values associate with other column values, for example "What else did buyers of x buy?", or "What projects did Sean, Ian and Gerard all work on?"

Start with buyers of x. The table that summarises this information might be a View that encapsulates joins from customers to orders to orderitems to products, perhaps scoped on a recent date range. Here we ignore all such detail. We focus only on the SQL patterns that solve this kind of problem:

```
DROP TABLE IF EXISTS userpurchases;
CREATE TABLE userpurchases( custID INT UNSIGNED, prodID INT UNSIGNED );
INSERT INTO userpurchases
VALUES (1,1),(1,2),(2,4),(3,1),(3,2),(4,2),(4,3),(5,1),(5,2),(5,3);
SELECT custID, GROUP_CONCAT(prodID ORDER BY prodID) AS PurchaseList
FROM userpurchases
GROUP BY custID;
+--------+--------------+
```

Read the entire item

Last updated 28 Jan 2011

Back to the top | Browse the book | Buy the book | Feedback

Join or subquery?

Usually, a `JOIN` is faster than an uncorrelated subquery. For example in the sakila test database, customer is a parent of rental (via customer_id) which in turn is a parent of payment (via rental_id). The subquery version of a query for whether a customer has made payments and rentals...

```
SELECT DISTINCT c.customer_id
FROM customer c
WHERE c.customer_id IN (
   SELECT r.customer_id
   FROM rental r
   JOIN payment p USING (rental_id)
   WHERE c.customer_id = 599;
);
```

is eight times slower than the join version...

```
SELECT DISTINCT c.customer_id
FROM customer c
JOIN rental r USING (customer_id)
JOIN payment p USING (rental_id)
WHERE c.customer_id = 599;
```

Running `EXPLAIN` on the two queries reveals why: the subquery version has to read most customer rows, while the join version proceeds inside out and discovers it needs to read just one customer row.

Last updated 22 May 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

Parents without children

Given tables `parent(id INT), child(id INT,parent_id INT)`, how do we find parents with no children? It's the *All X for which there is no Y* pattern, which can be written as an *exclusion join*...

```
SELECT parent.id
FROM parent
LEFT JOIN child ON parent.id = child.parent_id
WHERE child.parent_id IS NULL;
```

or with a `NOT EXISTS` subquery, which is logically equivalent to the exclusion join, but usually performs much slower:

```
SELECT parent.id AS ParentID
FROM parent
WHERE NOT EXISTS (
  SELECT parent.id
  FROM parent
  JOIN child ON parent.ID = child.parent_id
);
```

Last updated 28 Jan 2011

Back to the top *Browse the book* *Buy the book* *Feedback*

Parties who have contracts with one another

You have a parties table that holds info on peoples' names etc, and a contracts table where each row has clientID and contractorID value pointing at a parties.partyID value--that is, each contracts row points at two parties rows. You want to list the names of all contractors and their clients.

```
SELECT clientpartyID,
       pCli.name AS Client,
       contractorpartyID,
       pCon.name AS Contractor
FROM contracts
  INNER JOIN parties AS pCli
    ON contracts.clientpartyID = pCli.partyID
  INNER JOIN parties AS pCon
```

```
    ON contracts.contractorpartyID = pCon.partyID;
```

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

The unbearable slowness of IN()

You track orders and their items in `orders` and `orderdetails` tables, as in the NorthWind database. How many of your orders have been for multiple items? We can use the standard SQL `IN()` operator to answer the question:

```
SELECT orderID
FROM orders
WHERE orderID IN (
  SELECT orderID
  FROM orderdetails
  GROUP BY orderID
  HAVING COUNT(orderID) > 1
);
```

Read the entire item

Last updated 05 Nov 2015

Back to the top | Browse the book | Buy the book | Feedback

The [Not] Exists query pattern

Given a table `employee( employeeID INT, mgr_employeeID INT, salary DECIMAL(10,2))`, find the managers who earn less than one or more of their subordinates.

We can write this query directly from the logic of its spec...

```
SELECT DISTINCT employeeID
FROM employee AS e
WHERE EXISTS (
  SELECT employeeID
  FROM employee AS m
  WHERE m.mgr_employeeID = e.employeeID AND e.salary > m.salary
);
```

Read the entire item

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

What exams did a student not register for?

We have a `students` table, an `exams` table, and a `registrations` table in which each row registers one student for one exam. How do we find the exams for which a particular student is *not* registered?

```
DROP TABLE IF EXISTS students, exams, registrations;
CREATE TABLE students (
  sid int(10) unsigned PRIMARY KEY auto_increment,
  firstname varchar(45) NOT NULL default '',
  lastname varchar(45) NOT NULL default ''
);
INSERT INTO students VALUES
(1, 'Jack', 'Malone'),(2, 'Hiro', 'Nakamura'),(3, 'Bree', 'Van de Kamp'),
(4, 'Susan', 'Mayer'),(5, 'Matt', 'Parkman'),(6, 'Claire', 'Bennet');

CREATE TABLE exams (
```

Read the entire item

Last updated 28 Jan 2011

Back to the top | Browse the book | Buy the book | Feedback

List NULLs at end of query output

If ordering by `col`...

```
... ORDER BY IF(col IS NULL, 0, 1 ), col ...
```

Last updated 30 Dec 2009

Back to the top | Browse the book | Buy the book | Feedback

Parents with and without children

You have parties and contracts tables. Every contracts row has a contractorpartyID value which references a row in parties, and a clientpartyID value which also references a row in parties. How to list all parties and their contracts, showing blanks as empty strings rather than NULLs?

```
SELECT parties.partyID,
       IFNULL(contractorpartyID,'') AS contractor,
       IFNULL(clientpartyID,'') AS client
FROM parties
LEFT JOIN contractor_client ON partyID=contractorpartyID
ORDER BY partyID;
+---------+------------+--------+
| partyID | contractor | client |
+---------+------------+--------+
|       1 |            |        |
|       2 | 2          | 1      |
|       3 |            |        |
+---------+------------+--------+
```

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Emulate Row_Number()

ISO SQL defines a `ROW_NUMBER() OVER` function with an optional `PARTITION` clause for generating a derived row number column in a resultset. Several RDBMSs—including DB2, Oracle and SQL Server—implement it. Here is the simplest possible example. Given a table with two columns i and j, generate a resultset that has a derived sequential row_number column taking the values 1,2,3,... for a defined ordering of j which resets to 1 when the value of i changes:

```
DROP TABLE IF EXISTS test;
CREATE TABLE test(i int,j int);
INSERT INTO test
VALUES (3,31),(1,11),(4,14),(1,13),(2,21),(1,12),(2,22),(3,32),(2,23),(3,33);
```

The result must look like this:

Read the entire item

Last updated 29 Jan 2013

Back to the top | Browse the book | Buy the book | Feedback

Next row

You have a table of names, you have retrieved a row with name `$name`, and you want the row for the next name in name order. MySQL `LIMIT` syntax makes this very easy:

```
SELECT *
FROM tbl
WHERE name > $name
ORDER BY name
LIMIT 1
```

Last updated 22 May 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

Order by leading digits

To have column values 1abc,10abc,8abc appear in the expected order `1abc,8abc,10abc`, take advantage of a trick built into MySQL string parsing ...

```
SELECT '3xyz'+0;
+----------+
| '3xyz'+0 |
+----------+
|        3 |
+----------+
```

to write ...

```
SELECT ...
...
ORDER BY colname+0, colname;
```

Last updated 31 Mar 2012

Back to the top *Browse the book* *Buy the book* *Feedback*

Order by month name

The MySQL FIELD(str,str1,str2,...,strN) function returns 1 if str=str1, 2 if str=str2, etc., so ...

```
SELECT .
ORDER BY FIELD(month,'JAN','FEB','MAR',...,'NOV','DEC') .
```

will order query output from a legacy table in month-number order.

Last updated 22 May 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

Order by numerics then alphas

Given a column `type` with values 1,2,3,a,b,c, how to get the order 3,2,1,c,b,a?

```
ORDER BY type RLIKE '^[0-9]+$' DESC, `type` DESC
```

Last updated 22 May 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

Pagination

Suppose you have a phone book of names, addresses, etc. You are displaying 20 rows per page, you're on page 100, and you want to display page 99. How do you do this knowing only what page you are on?

Assuming...

- 1-based page numbers
- you are on page P
- each page shows N rows

then the general formula for translating a 1-based page number into a first `LIMIT` argument is ...

```
MAX(0,P-1) * N
```

which for the 99th 20-row page evaluates to 1960, and the second argument to `LIMIT` is just N, so to see page 99, write...

```
SELECT ... LIMIT (1960, N);
```

The trouble with this is scaling. MySQL doesn't optimise `LIMIT` at all well. `SELECT ... LIMIT 1000000,20` will unfortunately retrieve not just the twenty rows starting at the millionth row; it will retrieve a million and twenty rows before it shows you the 20 you asked for! The bigger the result, the longer `LIMIT` takes.

What's the alternative? Build pagination logic into the `WHERE` clause, and ensure that there is a covering index for the paginating column. On a table of 100,000 indexed random integers, `SELECT ... WHERE ...` for the last 20 integers in the table is twice as fast as the comparable `LIMIT` query. With a million integers, it's more than 500 times faster!

If your interface calls for showing only 20 rows per page on a given order, retrieve the twenty rows, *plus* the row just before the set if it exists, *plus* the next row after those twenty if it exists. When the user clicks the *Previous Page* button, adjust the `WHERE` clause to specify rows where the key value is <= the row just before the current set and `ORDER BY` the current index `DESC LIMIT 20`; likewise when the user clicks the *Next Page* button, have the `WHERE` clause specify key values >= that of the row just after the set, and `ORDER BY` the current index `ASC LIMIT 20`.

Last updated 22 Oct 2014

Back to the top *Browse the book* *Buy the book* *Feedback*

Suppress repeating ordering values

You have tables tracking authors and their books, for example:

```
CREATE TABLE author (
  id int(4) NOT NULL auto_increment PRIMARY KEY,
  name text NOT NULL
);
INSERT INTO author (id, name)
VALUES (1,'Brad Phillips'),(2,'Don Charles'),(3,'Kur Silver');
CREATE TABLE book (
  id int(4) NOT NULL auto_increment PRIMARY KEY,
  name text NOT NULL
);
INSERT INTO book (id, name)
```

Read the entire item

Last updated 22 May 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

All possible recipes with given ingredients

We have tables for recipes (r), ingredients required for recipes (ri), and ingredients now available in the pantry (p). In table `p` there may be many rows for a given recipe, each specifying one ingredient.

```
drop table if exists r,p,ri;
create table r(id int);
insert into r values(1),(2),(3);
create table p(id int);
insert into p values(1),(2),(3);
create table ri(rid int,pid int);
insert into ri values (1,1),(1,2),(2,1),(2,4),(3,5),(3,6),(3,7);
select id as recipes from r;
+---------+
| recipes |
```

```
+---------+
```

Read the entire item

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

All X for which all Y are Z (relational division)

You have an election database with tables listing political *parties*, election *districts*, and *candidates* running for parties in those districts. You want to know which parties have candidates running in all districts. Under **Aggregates** we show a `GROUP BY` solution (here).

If there are reasons not to aggregate, *relational division* can solve the problem. The basic idea in relational division is that, aside from aggregation, SQL has no direct way to express "all Xs for which all Y are Z", but does have a `NOT EXISTS` operator, so we can express "all Xs for which all Y are Z" in SQL as a double negative: "all Xs for which no Y is not Z". Once you think of formulating the question this way, the query almost writes itself:

```
SELECT DISTINCT party FROM parties
WHERE NOT EXISTS (
  SELECT * FROM districts
  WHERE NOT EXISTS (
    SELECT * FROM candidates
    WHERE candidates.party=parties.party AND candidates.district=districts.district
  )
);
```

Read the entire item

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Who makes all the parts for a given assembly?

One way to arrive at the answer is by asking: What are the assembly-supplier pairs such that no part of the assembly is not made by the supplier? That's relational division again, formulated for two tables by Stephen Todd. Given `assemblyparts(assembly,part)` and `partsuppliers(part,supplier)` tables, here is a query that Joe Celko credits to Pierre Mullin.

```
SELECT DISTINCT
  AP1.assembly,
  SP1.supplier
FROM AssemblyParts AS AP1, PartSuppliers AS SP1
WHERE NOT EXISTS (
  SELECT *
  FROM AssemblyParts AS AP2
  WHERE AP2.assembly = AP1.assembly
  AND NOT EXISTS (
    SELECT SP2.part
    FROM PartSuppliers AS SP2
    WHERE SP2.part = AP2.part AND SP2.supplier = SP1.supplier
  )
);
```

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Make a table of sequential ints

For example, a table of 1,000 ints starting from 0 with interval=1...

```
drop table if exists ints;
create table ints(i tinyint);
```

```
insert into ints values(0),(1),(2),(3),(4),(5),(6),(7),(8),(9);

drop table if exists temp;
create table temp
select 100*c.i+10*b.i+a.i as iseq
from ints a,ints b, ints c order by iseq;
```

Read the entire item

Last updated 21 Oct 2014

Back to the top | Browse the book | Buy the book | Feedback

Find adjacent unbooked theatre seats

A theatre booking service is often asked to book adjecent seats:

```
drop table if exists seats;
create table seats(row char(1),seat int,booked tinyint);
insert into seats values
('i',1,0),('i',2,0),('i',3,0),('i',4,0),('i',5,0),('i',6,0),('i',7,0),('i',8,0),
('i',9,0),('i',10,0),('i',11,0),('i',12,0),('i',13,0),('i',14,0),('i',15,0),
('j',1,1),('j',2,0),('j',3,1),('j',4,0),('j',5,0),('j',6,0),('j',7,1),('j',8,0),
('j',9,0),('j',10,0),('j',11,0),('j',12,0),('j',13,1),('j',14,0),('j',15,0);
```

The simplest method is a self-join:

Read the entire item

Last updated 26 Oct 2010

Back to the top | Browse the book | Buy the book | Feedback

Find average row-to-row time interval

You have a table of sequential times, and you're asked to report the average time interval between them in seconds.

Assuming table `t` with timestamp column `ts` and no `ts` duplicate values, the mean time difference between consecutive rows is the difference between the smallest (earliest) and largest (latest) timestamp divided by the rowcount - 1:

```
Select TimeStampDiff(SECOND, Min(ts), Max(ts) ) / ( Count(DISTINCT(ts)) -1 )
FROM t
```

Not sure of the formula?

1. The mean distance between consecutive rows is the sum of distances between consecutive rows, divided by the number of consecutive rows.

2. The sum of differences between consecutive rows is just the distance between the first row and last sorted row (assuming they are sorted by timestamp).

3. The number of consecutive rows is the total number of rows - 1.

Last updated 20 Jun 2014

Back to the top | Browse the book | Buy the book | Feedback

Find blocks of unused numbers

In a table of sequential IDs with no missing values, some are used and some are not. Find the blocks of unused IDs, if any:

```
DROP TABLE IF EXISTS tbl;
CREATE TABLE tbl(id INT,used BOOL);
INSERT INTO tbl VALUES (1,1),(2,1),(3,0),(4,1),(5,0),(6,1),(7,1),(8,1),
                       (9,0),(10,0),(11,1),(12,1),(13,0),(14,0),(15,0);
SELECT * FROM tbl;
```

```
+------+------+
| id   | used |
+------+------+
|    1 |    1 |
|    2 |    1 |
|    3 |    0 |
```

Read the entire item

Last updated 25 Sep 2009

Back to the top | Browse the book | Buy the book | Feedback

Find missing numbers in a sequence

You have a table `tbl(id int)` with values (1,2,4,18,19,20,21), and you wish to find the first missing number in its sequence of id values:

```
SELECT t1.id+1 AS Missing
FROM tbl AS t1
LEFT JOIN tbl AS t2 ON t1.id+1 = t2.id
WHERE t2.id IS NULL
ORDER BY id LIMIT 1;
+---------+
| Missing |
+---------+
|       3 |
+---------+
```

Read the entire item

Last updated 05 Mar 2015

Back to the top | Browse the book | Buy the book | Feedback

Find missing values in a range

You have a table named `tbl` with an integer primary key named `id`, and you need to know what key values between 0 and 999999 are missing in the table.

The simplest solution is an *exclusion join* from a virtual table of sequential numbers to the target table:

```
create or replace view v as
  select 0 i union select 1 union select 2 union select 3 union select 4
  union select 5 union select 6 union select 7 union select 8 union select 9;

select x.i
from (
  select a.i*100000 + b.i*10000 + c.i*1000 + d.i*100 + e.i*10 + f.i as i
  from v a
  join v b
  join v c
  join v d
  join v e
  join v f
) x
left join tbl on x.i=tbl.id
where tbl.id is null
order by x.i;
```

Last updated 11 Jul 2012

Back to the top | Browse the book | Buy the book | Feedback

Find previous and next values in a sequence

Given a table `t(ID int, ...)`, how would we display each ID and its next highest value?

A simple method ...

```
SELECT id, (SELECT MIN(id) from t as x WHERE x.id > t.id) AS Next
FROM t
ORDER BY id;
```

...but early versions of MySQL did not optimise correlated qubqueries at all well.

Read the entire item

Last updated 16 Dec 2015

Back to the top | Browse the book | Buy the book | Feedback

Find row with next value of specified column

Sometimes we need next values of a column on a given row ordering. Oracle has a `LEAD(...) OVER(...)` construct to simplify this query. MySQL does not.

A few efficient ways to do this are described under "Find previous and next values in a sequence". Here we look at more ambitious solutions.

The logic is:

1. Form a resultset consisting of all relevant rows, joined with all relevant rows that have greater values in the ordering columns. For example, if the table has these rows:

```
+------+
|    2 |
```

Read the entire item

Last updated 17 Apr 2015

Back to the top | Browse the book | Buy the book | Feedback

Find sequence starts and ends

A traditional way to find the first and last values of column value sequences in a table like this ...

```
drop table if exists t;
create table t(id int);
insert into t values(1),(2),(3),(4),(6),(7),(8);
```

... uses an exclusion join on the previous sequential value to find the first value of each sequence, and the minimum next value from a left join and an exclusion join on the previous sequential value to find the end of each sequence:

```
SELECT
```

Read the entire item

Last updated 14 Dec 2014

Back to the top | Browse the book | Buy the book | Feedback

Find shortest & longest per-user event time intervals

You have a table of users and event times, and you need to find the shortest and longest per-user event time intervals:

```
drop table if exists t;
create table t( t timestamp, user smallint);
insert into t values
('2014-11-28 18:30:02',  1),('2014-11-28 18:30:05',  1),('2014-11-
28 18:30:08',  1),
('2014-11-28 18:30:11',  1),('2014-11-28 18:30:15',  1),('2014-11-
28 18:30:18',  1),
```

```
('2014-11-28 18:30:21',  1),('2014-11-28 18:30:23',  1),('2014-11-
28 18:30:26',  1),
('2014-11-28 18:30:29',  2),('2014-11-28 18:30:32',  2),('2014-11-
28 18:30:33',  2),
('2014-11-28 18:30:37',  2),('2014-11-28 18:30:40',  2),('2014-11-
28 18:30:42',  2),
('2014-11-28 18:30:44',  2),('2014-11-28 18:31:01',  2),('2014-11-
28 18:31:04',  2),
('2014-11-28 18:31:07',  2),('2014-11-28 18:31:10',  2);
```

Read the entire item

Last updated 03 Dec 2014

Back to the top | Browse the book | Buy the book | Feedback

Find specific sequences

You have a table which tracks hits on named web pages...

```
CREATE TABLE hits (
  id INT NOT NULL DEFAULT 0,
  page CHAR(128) DEFAULT '',
  time TIMESTAMP NOT NULL DEFAULT 0,
  PRIMARY KEY(id, time)
)
```

where id is unique to a session. Here is a bit of sample data:

Read the entire item

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Find the next value after a sequence

Given a sequence of values on a given ordering, what is the next value? It's a common requirement (eg in DNA sequencing). A MySQL Forum user posted this example:

```
drop table if exists data;
create table data(id smallint unsigned primary key auto_increment, val smallint);
insert into data (val) values
(8),(21),(28),(29),(31),(32),(27),(20),(31),(1),(18),(35),
(18),(30),(22),(9),(2),(8),(33),(8),(19),(31),(6),(31),(14),(5),
(26),(29),(34),(34),(19),(27),(29),(3),(21),(18),(31),(5),(18),
(34),(4),(15),(12),(20),(28),(31),(13),(22),(19),(30),(0),(2),
(30),(28),(2),(10),(27),(9),(23),(28),(29),(16),(16),(31),(35),(18),
(2),(15),(1),(30),(15),(11),(17),(26),(35),(1),(22),(19),(23),(1),
(18),(35),(28),(13),(9),(14);
```

Read the entire item

Last updated 31 Mar 2012

Back to the top | Browse the book | Buy the book | Feedback

Gaps in a time series

Advanced time series analysis generally requires custom software, but straightforward SQL queries can answer simple time series questions. You have a jobtimes table with columns ID, job, machine, start_time, and stop_time. You wish to know which machines have had gaps between activity periods. It's a version of "Find available booking periods":

```
drop table jobtimes;
create table jobtimes(id int, machine smallint, start_time timestamp, stop_time timestamp);
insert into jobtimes values(1,1,'2011-7-1 08:00:00', '2011-7-1 10:00:00');
insert into jobtimes values(2,1,'2011-7-1 11:00:00', '2011-7-1 14:00:00');
insert into jobtimes values(3,2,'2011-7-1 08:00:00', '2011-7-1 09:00:00');
```

```
insert into jobtimes values(4,2,'2011-7-1 09:00:00', '2011-7-1 10:00:00');
insert into jobtimes values(5,3,'2011-7-1 08:00:00', '2011-7-1 08:30:00');
insert into jobtimes values(6,3,'2011-7-1 10:00:00', '2011-7-1 12:00:00');
select * from jobtimes;
+------+---------+---------------------+---------------------+
| id   | machine | start_time          | stop_time           |
```

Read the entire item

Last updated 15 Jul 2011

Back to the top *Browse the book* *Buy the book* *Feedback*

Make values of a column sequential

You have a table *tbl* with a column whose values can be freely overwritten and which needs to be filled with perfectly sequential values starting with 1:

```
SET @i=0;
UPDATE tbl SET keycol=(@i:=@i+1);
```

But more often, what you need is a mechanism for guaranteeing an unbroken sequence of integer key values whenever a row is inserted.

Here is the bad news: `auto_increment` does not provide this logic, for example it does not prevent inserton failures, editing of the column, or subsequent deletion.

The good news is that the logic for guaranteeing an unbroken sequence is straightforward:

(i) use InnoDB

(ii) create a table with the desired sequence in one column and further column(s) to track use (to use such a table for sequential columns in multiple other tables, provide a tracking column for each such table)

(iii) write Triggers to fetch the next available sequential key from that table, mark it used, and assign it on Insert

(iv) put all such logic inside transaction blocks

(v) prohibit deletion in the table.

Last updated 05 Mar 2013

Back to the top *Browse the book* *Buy the book* *Feedback*

Track stepwise project completion

A master table has one row for each project, and the number of sequential steps required to complete each project. A detail table has one row per project per completed step:

```
DROP TABLE IF EXISTS t1 ;
CREATE TABLE t1 (
  id INT, projectname CHAR(2), projectsteps INT
);
INSERT INTO t1 VALUES
(1, 'xx', 3),
(2, 'yy', 3),
(3, 'zz', 5);

DROP TABLE IF EXISTS t2;
CREATE TABLE t2 (
```

Read the entire item

Last updated 22 May 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

Winning Streaks

Given a table of IDs and won-lost results, how do we find the longest winning streak?

```
drop table if exists results;
create table results(id int,result char(1));
insert into results values
(1,'w'),(2,'l'),(3,'l'),(4,'w'),(5,'w'),(6,'w'),(7,'l'),(8,'w'),(9,'w');
select * from results;
+------+--------+
| id   | result |
+------+--------+
|    1 | w      |
|    2 | l      |
|    3 | l      |
```

Read the entire item

Last updated 22 May 2009

Back to the top Browse the book Buy the book Feedback

Great circle distance

Find the distance in kilometres between two points on the surface of the earth. This is just the sort of problem stored functions were made for. For a first order approximation, ignore deviations of the earth's surface from the perfectly spherical. Then the distance in radians is given by a number of trigonometric formulas. ACOS and COS behave reasonably:

```
             COS(lat1-lat2)*(1+COS(lon1-lon2)) - COS(lat1+lat2)*(1-COS(lon1-
lon2))
rads = ACOS( ------------------------------------------------------------------
-- )
                                          2
```

We need to convert degrees latitude and longitude to radians, and we need to know the length in km of one radian on the earth's surface, which is 6378.388. The function:

```
set log_bin_trust_function_creators=TRUE;
```

Read the entire item

Last updated 22 May 2009

Back to the top Browse the book Buy the book Feedback

Moving average

Given a table of dates and daily values, retrieve their moving 5-day average:

```
DROP TABLE IF EXISTS t;
CREATE TABLE t (dt DATE, qty INT);
INSERT INTO t VALUES
('2007-1-1',5),('2007-1-2',6),('2007-1-3',7),('2007-1-4',8),('2007-1-5',9),
('2007-1-6',10),('2007-1-7',11),('2007-1-8',12),('2007-1-9',13);

SELECT
  a.dt,
  a.qty,
  Round( ( SELECT SUM(b.qty) / COUNT(b.qty)
           FROM t AS b
```

Read the entire item

Last updated 16 Jul 2012

Back to the top Browse the book Buy the book Feedback

Multiple sums across a join

You have a *parties* table that holds info on peoples' names etc, and a *contracts* table in which each row defines one contract, identifying a client as clientpartyID and a contractor as contractorpartyID, each of these a foreign key referencing parties.partyID. You want a list of parties showing how many contracts they have participated in as client, and how many they've participated in as contractor.

```
SELECT
  p.partyID,
  p.name,
  (SELECT COUNT(*) FROM contractor_client c1 WHERE c1.clientpartyID = p.partyID )
  AS ClientDeals,
  (SELECT COUNT(*) FROM contractor_client c2 WHERE c2.contractorpartyID = p.partyID)
  AS ContractorDeals
FROM parties p
ORDER BY partyID;
```

Last updated 22 May 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

Percentiles

In the Sakila table `film`, retrieve a top-down percentile ranking of film lengths:

```
SELECT
  a.film_id ,
  ROUND( 100.0 * ( SELECT COUNT(*) FROM film AS b WHERE b.length <= a.length ) / total.cnt, 1 )
  AS percentile
FROM film a
CROSS JOIN (
  SELECT COUNT(*) AS cnt
  FROM film
) AS total
ORDER BY percentile DESC;
```

Read the entire item

Last updated 07 Oct 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

Running sums, chequebooks

A user variable can maintain a per-row cumulative sum of column values. Initialise it, then adjust its value as desired in the appropriate `SELECT` expression:

```
SET @total=0;
SELECT id, value, @total:=@total+value AS RunningSum
FROM tbl;
```

Chequebook balancing programs often use this pattern. This one tracks the running balance and how much money from the most recent deposit remains:

```
DROP TABLE IF EXISTS chequebook;
```

Read the entire item

Last updated 06 Jan 2011

Back to the top *Browse the book* *Buy the book* *Feedback*

Sum across categories

You often need to sum across several categories to total customer purchase amounts, salesperson sales amounts, political party election spending, etc.

For this example assume three tables: candidates, parties and ridings. You want

to get the total amount spent in all ridings by every party in one output row. Here is the schema:

```
CREATE TABLE candidates (
  id int(11) NOT NULL default '0',
  `name` char(10) ,
  riding char(12) ,
  party char(12) ,
  amt_spent decimal(10,0) NOT NULL default '0',
  PRIMARY KEY  (id)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

Read the entire item

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Top ten

We often want to know the top 1, 2, 10 or whatever values from a query. This is dead simple in MySQL. However many JOINs and WHEREs the query has, simply ORDER BY the column(s) whose highest values are sought, and LIMIT the resultset:

```
  SELECT (somecolumn), (othercolumns) ...
  FROM (some tables) ...
  ORDER BY somecolumn DESC
  LIMIT 10;
```

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

A cursor if necessary, but not necessarily a cursor

You have `photos (id INT, photo BLOB, tally INT)` and `votes(id INT, userID INT, photoID INT)` tables. You wish to update `photos.tally` values from counts per photo in the `votes` table. You can use a cursor to walk the photos table, updating the tally as you go:

```
DROP TABLE IF EXISTS photos;
CREATE TABLE photos (id INT, photo BLOB, tally INT);
INSERT INTO photos VALUES(1,'',0),(2,'',0);
DROP TABLE IF EXISTS VOTES;
CREATE TABLE VOTES( userID INT, photoID INT);
INSERT INTO votes VALUES (1,1),(2,1),(2,2);

DROP PROCEDURE IF EXISTS updatetallies;
DELIMITER //
CREATE PROCEDURE updatetallies()
BEGIN
```

Read the entire item

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Emulate sp_exec

Sometimes it is desirable to call multiple stored procedures in one command. In SQL Server this can be done with `sp_exec`. In MySQL we can easily write such an sproc that calls as many sprocs as we please, for example...

```
USE sys;
DROP PROCEDURE IF EXISTS sp_exec;
DELIMITER |
CREATE PROCEDURE sp_exec( p1 CHAR(64), p2 CHAR(64) )
BEGIN
  -- permit doublequotes to delimit data
```

```
  SET @sqlmode=(SELECT @@sql_mode);
  SET @@sql_mode='';
  SET @sql = CONCAT( "CALL ", p1 );
  PREPARE stmt FROM @sql;
  EXECUTE stmt;
  DROP PREPARE stmt;
  SET @sql = CONCAT( "CALL ", p2 );
  PREPARE stmt FROM @sql;
  EXECUTE stmt;
  DROP PREPARE stmt;
  SET @@sql_mode=@sqlmode;
END;
|
DELIMITER ;
```

Last updated 22 May 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

Variable-length argument for query IN() clause

To have an sproc accept a variable-length parameter list for an `IN(...)` clause in a query, code the sproc to `PREPARE` the query statement:

```
DROP PROCEDURE IF EXISTS passInParam;
DELIMITER |
CREATE PROCEDURE passInParam( IN qry VARCHAR(100), IN param VARCHAR(1000) )
BEGIN
  SET @qry = CONCAT( qry, param, ')' );
  PREPARE stmt FROM @qry;
  EXECUTE stmt;
  DROP PREPARE stmt;
END;
|
DELIMITER ;
```

For this example, the query string should be of the form:

```
SELECT ... FROM ... WHERE ... IN (
```

but so long as it has those elements, it can be as complex as you like. When you call the sproc:
1. Quote each argument with a *pair* of single quotes,
2. Separate these quoted arguments with commas,
3. Surround the whole `param` string with another set of single quotes:

```
CALL passInParam( 'SELECT * FROM tbl WHERE colval IN (', ('''abc'',''def'',''ghi''' ));
```

Last updated 22 May 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

Count delimited substrings

Here is a function to count substrings delimited by a constant delimiting string:

```
DROP FUNCTION IF EXISTS strcount;
SET GLOBAL log_bin_trust_function_creators=1;
DELIMITER |
CREATE FUNCTION strCount( pDelim VARCHAR(32), pStr TEXT) RETURNS int(11)
BEGIN
  DECLARE n INT DEFAULT 0;
  DECLARE pos INT DEFAULT 1;
  DECLARE strRemain TEXT;
  SET strRemain = pStr;
  SET pos = LOCATE( pDelim, strRemain );
  WHILE pos != 0 DO
```

Read the entire item

Last updated 22 May 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

Count substrings

To count instances of a search string in a target string ...

- in the target string, replace the search string with a single character,
- subtract the length of the modified target string from the length of the original target string,
- divide that by the length of the search string:

```
SET @str  = "The quick brown fox jumped over the lazy dog";
SET @find = "the";
SELECT ROUND(((LENGTH(@str) - LENGTH(REPLACE(LCASE(@str), @find, '')))/LENGTH(@find)),0)
AS COUNT;
+-------+
| COUNT |
+-------+
|     2 |
+-------+
```

Note that REPLACE() does a case-sensitive search; to get a case-insensitive result you must coerce target and search strings to one case.

To remove decimals from the result:

```
SELECT CAST((LENGTH(@str) - LENGTH(REPLACE(LCASE(@str)), @find, '')))/LENGTH(@find) AS SIGNED) AS COUNT;
```

Last updated 22 May 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

Levenshtein distance

The Levenshtein distance between two strings is the minimum number of operations needed to transform one string into the other, where an operation may be insertion, deletion or substitution of one character. Jason Rust published this MySQL algorithm for it at http://www.codejanitor.com/wp/.

```
CREATE FUNCTION levenshtein( s1 VARCHAR(255), s2 VARCHAR(255) )
  RETURNS INT
  DETERMINISTIC
  BEGIN
    DECLARE s1_len, s2_len, i, j, c, c_temp, cost INT;
    DECLARE s1_char CHAR;
    -- max strlen=255
    DECLARE cv0, cv1 VARBINARY(256);
    SET s1_len = CHAR_LENGTH(s1), s2_len = CHAR_LENGTH(s2), cv1 = 0x00, j = 1, i = 1, c = 0;
    IF s1 = s2 THEN
      RETURN 0;
```

Read the entire item

Last updated 22 May 2009

Back to the top *Browse the book* *Buy the book* *Feedback*

Proper case

The basic idea is...

- lower-case the string
- upper-case the first character if it is a-z, and any other a-z character that follows a punctuation character

Here is the function. To make it work with strings long than 128 characters, change its input and return declarations accordingly:

```
DROP FUNCTION IF EXISTS proper;
SET GLOBAL  log_bin_trust_function_creators=TRUE;
DELIMITER |
```

```
CREATE FUNCTION proper( str VARCHAR(128) )
RETURNS VARCHAR(128)
BEGIN
  DECLARE c CHAR(1);
```

Read the entire item

Last updated 20 Jun 2012

Back to the top | Browse the book | Buy the book | Feedback

Retrieve octets from IP addresses

If `ip` is for the form `N.N.N.N` where N may be 1, 2 or 3 digits, how to group and count by just the first three octets, ie the ip class?

```
SELECT
  LEFT(ip, CHAR_LENGTH(ip) - LOCATE('.', REVERSE(ip))) as ipclass,
  COUNT(*)
FROM tbl
GROUP BY ipclass;
```

Hamilton Turner notes we can find the first octet with `LEFT(ip,LOCATE('.',ip)-1)`.

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Return digits or alphas from a string

Return only the digits from a string:

```
SET GLOBAL log_bin_trust_function_creators=1;
DROP FUNCTION IF EXISTS digits;
DELIMITER |
CREATE FUNCTION digits( str CHAR(32) ) RETURNS CHAR(32)
BEGIN
  DECLARE i, len SMALLINT DEFAULT 1;
  DECLARE ret CHAR(32) DEFAULT '';
  DECLARE c CHAR(1);
  SET len = CHAR_LENGTH( str );
  REPEAT
    BEGIN
```

Read the entire item

Last updated 22 May 2009

Back to the top | Browse the book | Buy the book | Feedback

Split a string

```
CREATE FUNCTION SPLIT_STR( s VARCHAR(255), delim VARCHAR(16),  pos INT )
RETURNS VARCHAR(255)
RETURN REPLACE( SUBSTRING( SUBSTRING_INDEX(s, delim, pos),
                           LENGTH(SUBSTRING_INDEX(s, delim, pos -1)) + 1),
                delim,
                ''
              );
```

For longer strings, change the datatype.

Last updated 27 Jan 2012

Back to the top | Browse the book | Buy the book | Feedback

Strip HTML tags

Ported from a T-SQL function by Robert Davis:

```
SET GLOBAL log_bin_trust_function_creators=1;
```

```
DROP FUNCTION IF EXISTS fnStripTags;
DELIMITER |
CREATE FUNCTION fnStripTags( Dirty varchar(4000) )
RETURNS varchar(4000)
DETERMINISTIC
BEGIN
  DECLARE iStart, iEnd, iLength int;
  WHILE Locate( '<', Dirty ) > 0 And Locate( '>', Dirty, Locate( '<', Dirty )) > 0 DO
    BEGIN
      SET iStart = Locate( '<', Dirty ), iEnd = Locate( '>', Dirty, Locate('<', Dirty ));
```

Read the entire item

Last updated 16 Jul 2012

Back to the top | Browse the book | Buy the book | Feedback