INFO1-CE9224: Introduction to PHP Programming

Session 8
August 8, 2012

Resources

http://davehauenstein.com/nyu/INFO1-CE9224-2012-Summer

Username: nyuscps
Password: $nyuscps$

Class 8 Agenda

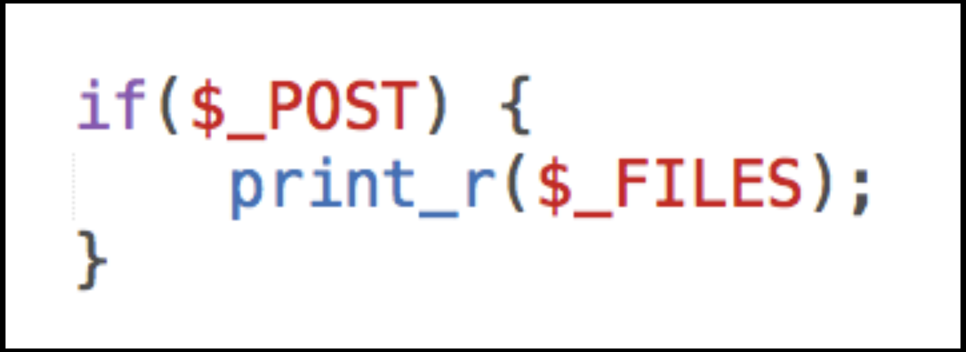
- Uploading Files
- Database Overview
- Relational Databases (RDBMS)
- Tables, Rows, Attributes, Constraints
- Normalization
- Primary and Foreign Keys
- SQL - Structured Query Language
- Retrieving, Storing, Updating Data
- Retrieving Related Data (Joins)
- Lab and Installing phpMyAdmin in MAMP

Uploading Files

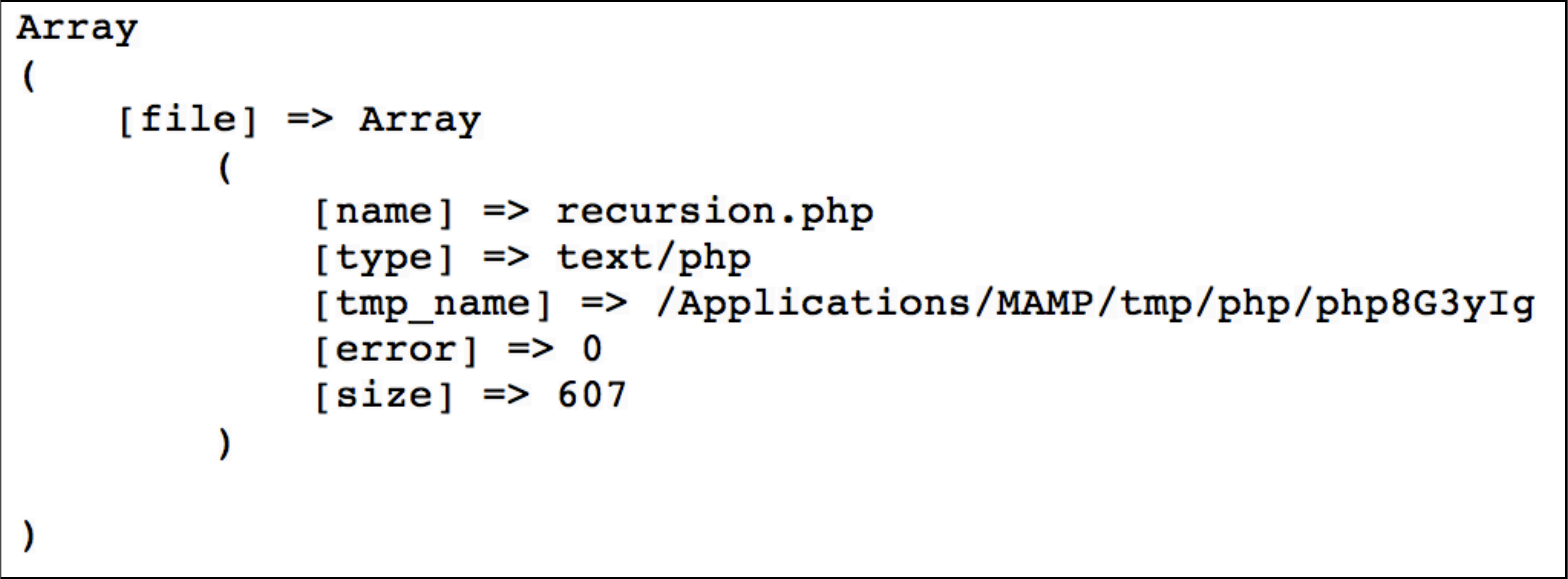
Uploading Files

- Set Up Form
  - <form enctype="multipart/form-data" ...>
  - <input type="file" name="a_file" id="a_file" />
- Use PHP to determine if file was successfully uploaded.
- If submitted without error, file is moved to a temporary place on the server.
- Use PHP to copy file from temporary location to permanent location.
- Use file for whatever purpose application needs it.

Uploading Files

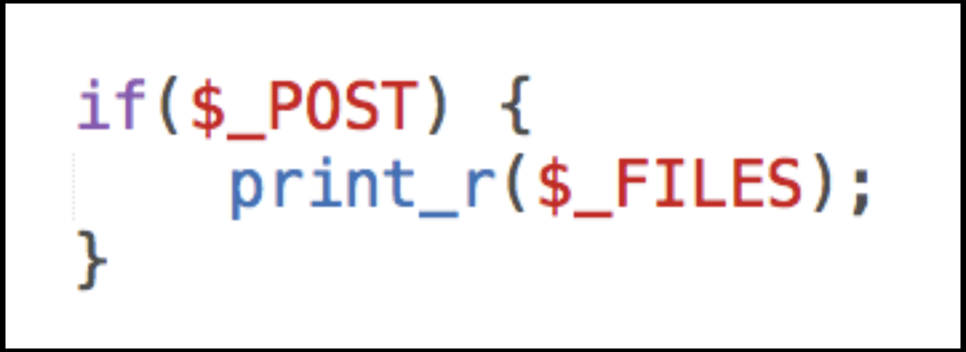


```php
if($_POST) {
    print_r($_FILES);
}
```

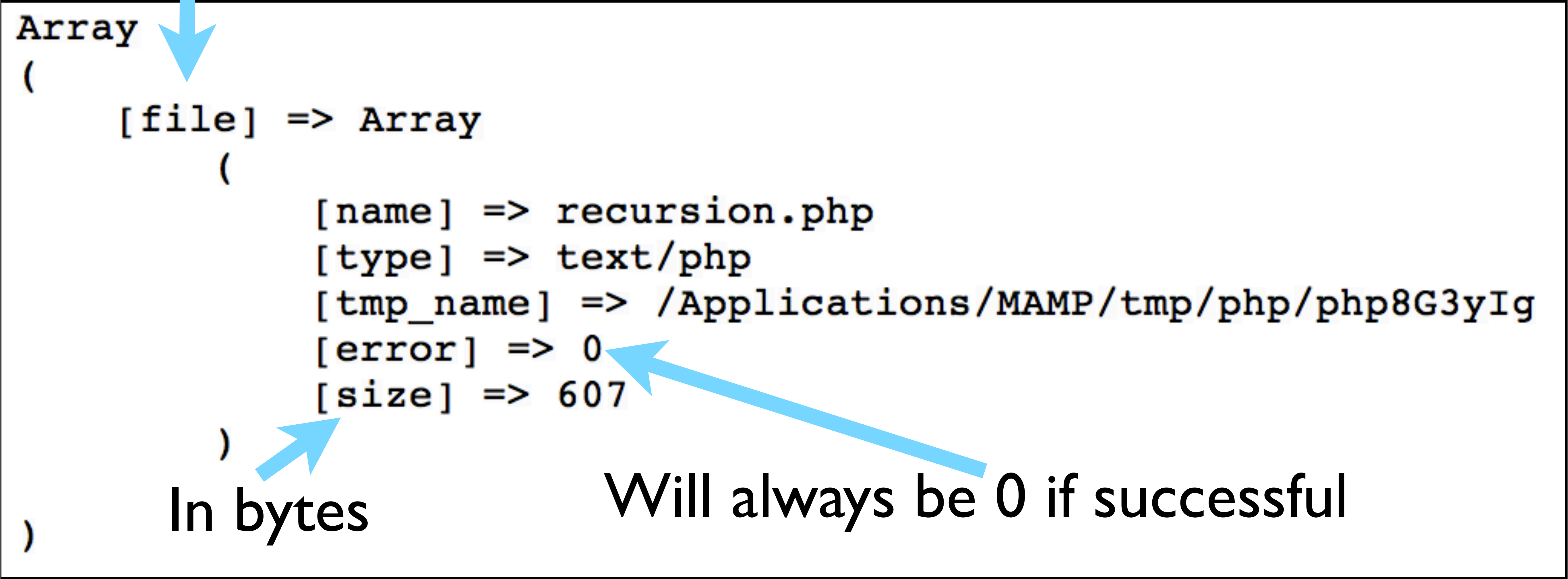


```
Array
(
    [file] => Array
        (
            [name] => recursion.php
            [type] => text/php
            [tmp_name] => /Applications/MAMP/tmp/php/php8G3yIg
            [error] => 0
            [size] => 607
        )

)
```

Uploading Files



```
if($_POST) {
    print_r($_FILES);
}
```



Name of form field

```
Array
(
    [file] => Array
        (
            [name] => recursion.php
            [type] => text/php
            [tmp_name] => /Applications/MAMP/tmp/php/php8G3yIg
            [error] => 0
            [size] => 607
        )

)
```

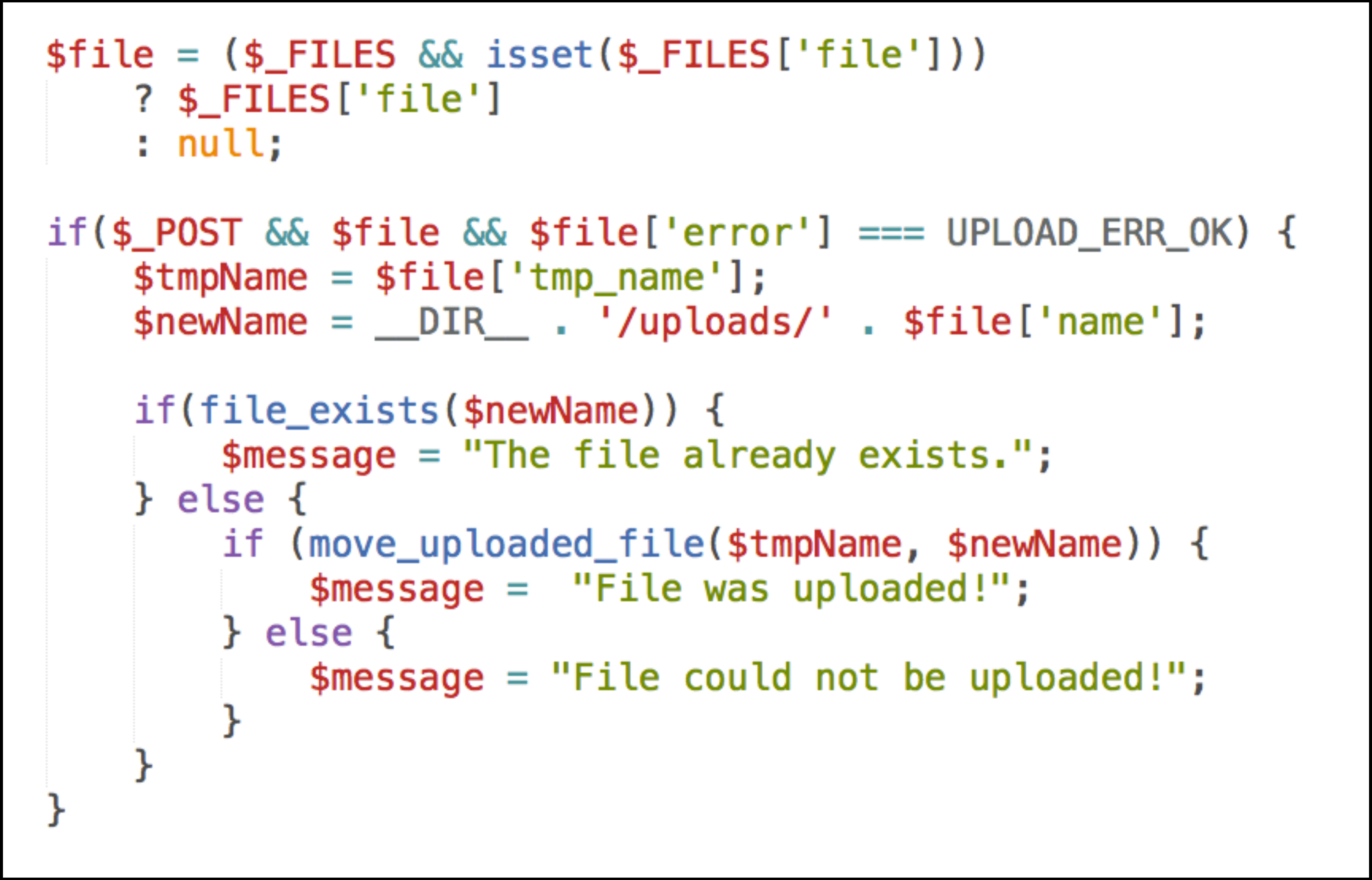
In bytes

Will always be 0 if successful

Errors

PHP Constant	Value	Description
UPLOAD_ERR_OK	0	There is no error, the file uploaded with success.
UPLOAD_ERR_INI_SIZE	1	The uploaded file exceeds the upload_max_filesize directive in *php.ini*
UPLOAD_ERR_FORM_SIZE	2	The uploaded file exceeds the MAX_FILE_SIZE directive that was specified in the HTML form.
UPLOAD_ERR_PARTIAL	3	The uploaded file was only partially uploaded.
UPLOAD_ERR_NO_FILE	4	No file was uploaded.
UPLOAD_ERR_NO_TMP_DIR	6	No file was uploaded.
UPLOAD_ERR_CANT_WRITE	7	Failed to write file to disk. Introduced in PHP 5.1.0.
UPLOAD_ERR_EXTENSION	8	A PHP extension stopped the file upload.

Uploading Files



```
$file = ($_FILES && isset($_FILES['file']))
    ? $_FILES['file']
    : null;

if($_POST && $file && $file['error'] === UPLOAD_ERR_OK) {
    $tmpName = $file['tmp_name'];
    $newName = __DIR__ . '/uploads/' . $file['name'];

    if(file_exists($newName)) {
        $message = "The file already exists.";
    } else {
        if (move_uploaded_file($tmpName, $newName)) {
            $message =  "File was uploaded!";
        } else {
            $message = "File could not be uploaded!";
        }
    }
}
```

Uploading Files Review

1. Check for POST
2. Check to see if $_FILES['your_file'] is set
3. Check to see if $_FILES['your_file']['error'] is equal to 0 or the constant UPLOAD_ERR_OK
4. Assume everything else is ok, copy the tmp file, to the new location of the uploaded file.
5. Additionally...
    1. check if file exists before overwriting
    2. Check that the destination folder exists and is writeable

Databases

- A place where programs can store data for retrieval at a later time.
- Think of a database as an electronic filing system.
- Data can be easily accessed, stored, managed, and updated.
- Endless use cases: Storing user accounts, products for an ecommerce platform, etc...

Why Not Use Files For a Datastore?

- This approach does not scale.
- Files lock, and multiple applications can not talk to them at the same time.
- Offload all of your processing to application code (aka PHP).
- Good for very small scale, localized tasks.
- This will all become apparent to you as you learn more about database applications.

Types of Databases

- Embedded Databases
- Client-Server Database
  - Key-Value
  - Relational (RDBMS)

Embedded Databases

- Ships with the application that uses it. Think iPhone app, for example.
- Is not networked; cannot be shared across servers by different applications.
- Fast, light weight, easy to configure and deploy.
- dbase, dbm SQLite

Key/Value Databases (NoSQL)

- New technology created to deal with massive data and real-time scaling problems. Think Google, Twitter, Facebook.
- Distributed Architecture.
- Most are “dumb”. Data integrity is not a first-class consideration.
- Structure is of minimal importance.
- MongoDb, CouchDb, Hadoop, Redis

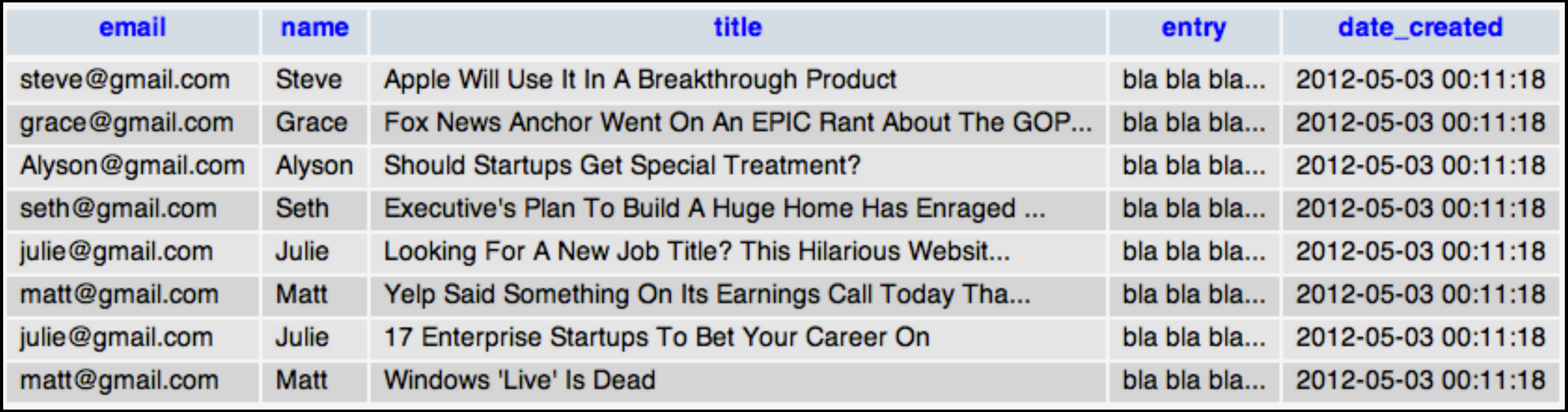
Relational Database (RDBMS)

- Data is stored and can be associated with other types of data based on common attributes. Think customers, products, orders, shopping cart.
- Data integrity is of the utmost importance.
- Data is stored in *tables* which have *rows* and *columns*.
- Microsoft SQL Server, Oracle, MySQL

Relational Database

- This is what we will cover in class.
- Most common approach for storing data for web applications.
- MySQL and PHP have always been very close friends.
- Cheap (free!), extremely powerful, widely available.

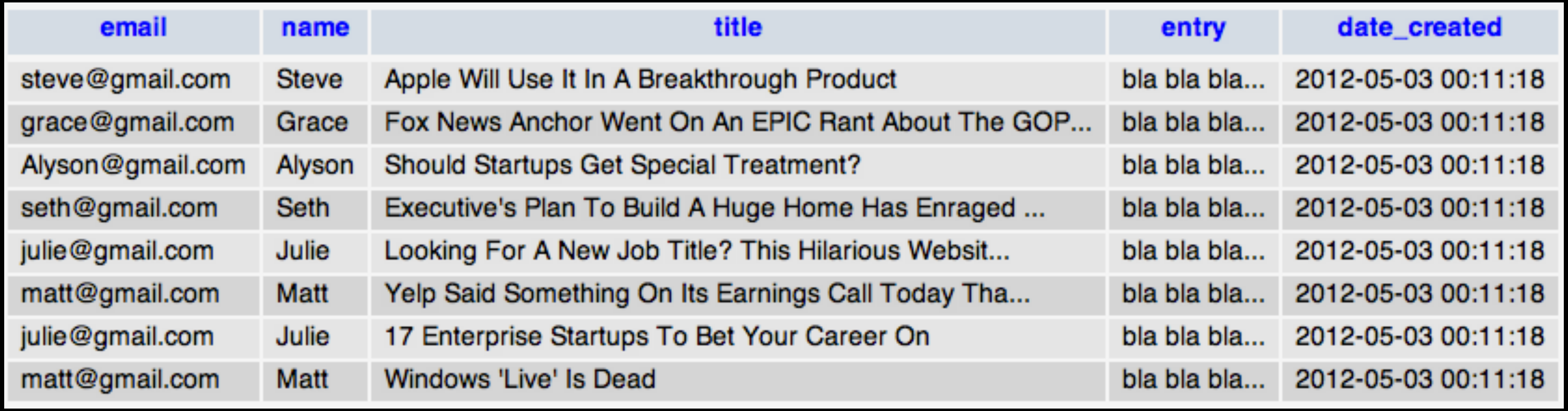
Example Database Table



email	name	title	entry	date_created
steve@gmail.com	Steve	Apple Will Use It In A Breakthrough Product	bla bla bla...	2012-05-03 00:11:18
grace@gmail.com	Grace	Fox News Anchor Went On An EPIC Rant About The GOP...	bla bla bla...	2012-05-03 00:11:18
Alyson@gmail.com	Alyson	Should Startups Get Special Treatment?	bla bla bla...	2012-05-03 00:11:18
seth@gmail.com	Seth	Executive's Plan To Build A Huge Home Has Enraged ...	bla bla bla...	2012-05-03 00:11:18
julie@gmail.com	Julie	Looking For A New Job Title? This Hilarious Websit...	bla bla bla...	2012-05-03 00:11:18
matt@gmail.com	Matt	Yelp Said Something On Its Earnings Call Today Tha...	bla bla bla...	2012-05-03 00:11:18
julie@gmail.com	Julie	17 Enterprise Startups To Bet Your Career On	bla bla bla...	2012-05-03 00:11:18
matt@gmail.com	Matt	Windows 'Live' Is Dead	bla bla bla...	2012-05-03 00:11:18

- This is a representation of a database table.
- Each row represents a blog entry by some author.
- Each column contains a specific type of data about each blog entry.
- The blue text in the top row is the *column name*.

Problems



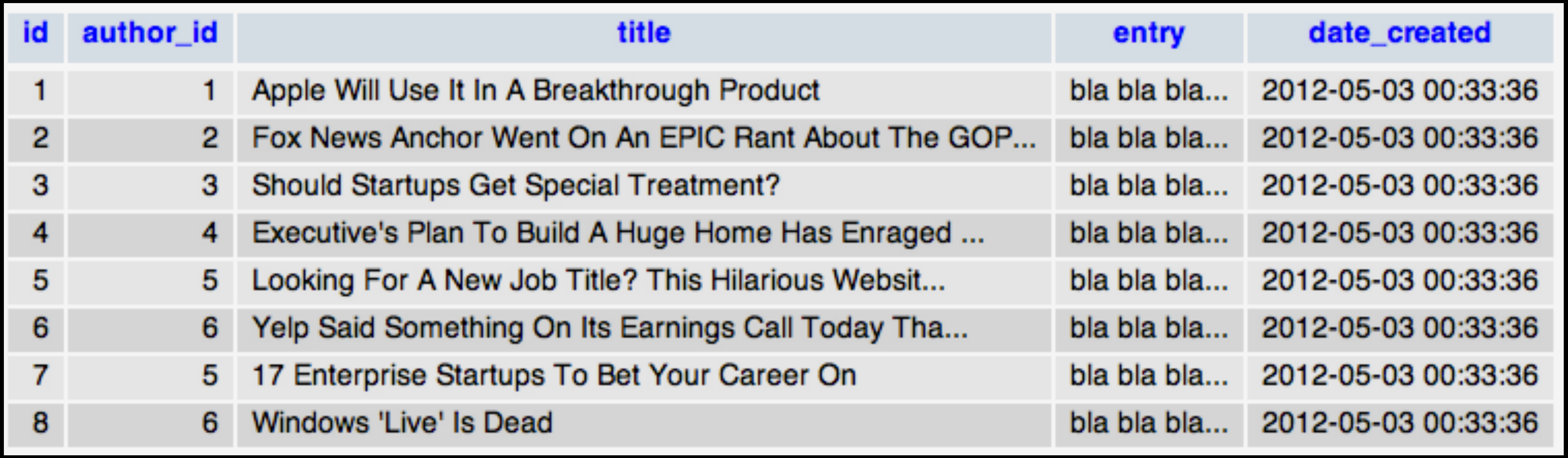
email	name	title	entry	date_created
steve@gmail.com	Steve	Apple Will Use It In A Breakthrough Product	bla bla bla...	2012-05-03 00:11:18
grace@gmail.com	Grace	Fox News Anchor Went On An EPIC Rant About The GOP...	bla bla bla...	2012-05-03 00:11:18
Alyson@gmail.com	Alyson	Should Startups Get Special Treatment?	bla bla bla...	2012-05-03 00:11:18
seth@gmail.com	Seth	Executive's Plan To Build A Huge Home Has Enraged ...	bla bla bla...	2012-05-03 00:11:18
julie@gmail.com	Julie	Looking For A New Job Title? This Hilarious Websit...	bla bla bla...	2012-05-03 00:11:18
matt@gmail.com	Matt	Yelp Said Something On Its Earnings Call Today Tha...	bla bla bla...	2012-05-03 00:11:18
julie@gmail.com	Julie	17 Enterprise Startups To Bet Your Career On	bla bla bla...	2012-05-03 00:11:18
matt@gmail.com	Matt	Windows 'Live' Is Dead	bla bla bla...	2012-05-03 00:11:18

- This design is inefficient. For every entry, the author's name and email address are repeated.
- Redundancy is a waste of space on the database server's hard drive and processing time.
- What if author's email address changes? Must update every single row.

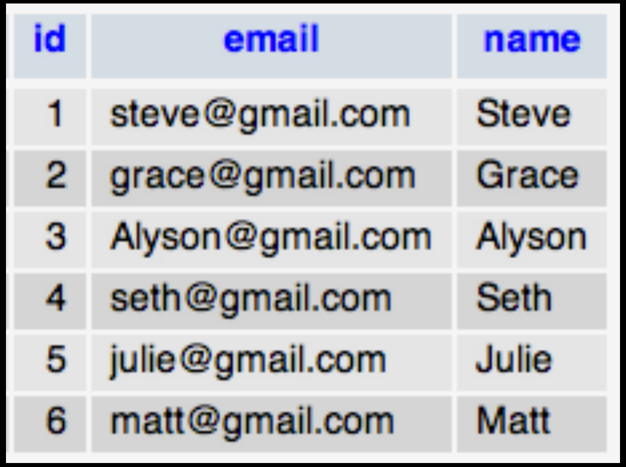
Solution: Normalization

- Break up *denormalized* tables into several *normalized* tables to avoid redundancy.
- What repeats? Author information. It is a one-many-relationship. One author has many entries, or, many entries can have the same author.
- Use a unique identifier to represent an author in the entries table.

Solution: Normalization



id	author_id	title	entry	date_created
1	1	Apple Will Use It In A Breakthrough Product	bla bla bla...	2012-05-03 00:33:36
2	2	Fox News Anchor Went On An EPIC Rant About The GOP...	bla bla bla...	2012-05-03 00:33:36
3	3	Should Startups Get Special Treatment?	bla bla bla...	2012-05-03 00:33:36
4	4	Executive's Plan To Build A Huge Home Has Enraged ...	bla bla bla...	2012-05-03 00:33:36
5	5	Looking For A New Job Title? This Hilarious Websit...	bla bla bla...	2012-05-03 00:33:36
6	6	Yelp Said Something On Its Earnings Call Today Tha...	bla bla bla...	2012-05-03 00:33:36
7	5	17 Enterprise Startups To Bet Your Career On	bla bla bla...	2012-05-03 00:33:36
8	6	Windows 'Live' Is Dead	bla bla bla...	2012-05-03 00:33:36

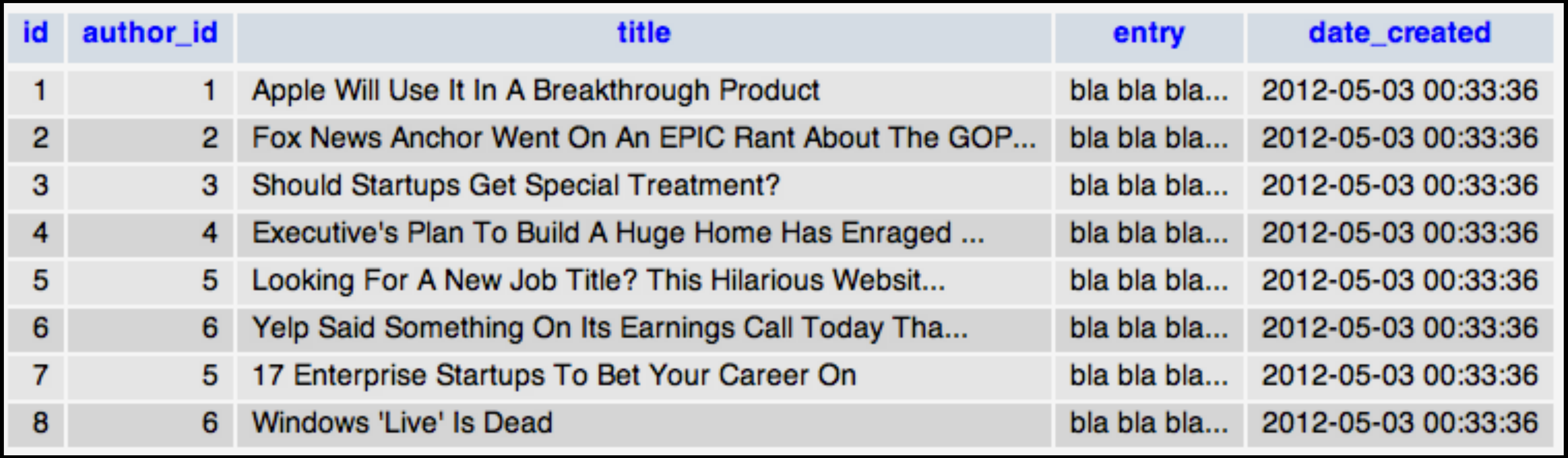


id	email	name
1	steve@gmail.com	Steve
2	grace@gmail.com	Grace
3	Alyson@gmail.com	Alyson
4	seth@gmail.com	Seth
5	julie@gmail.com	Julie
6	matt@gmail.com	Matt

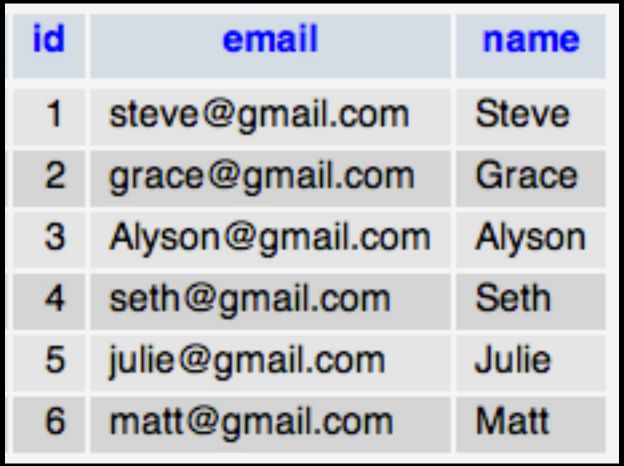
Solution: Normalization

- Authors table has each author *only one time* with their respective information.
- A new column has been added to entries, *author_id*, which references the *id* column in the Authors table.
- The id column in the Authors table is a unique identifier called the *primary key*.

Solution: Normalization



id	author_id	title	entry	date_created
1	1	Apple Will Use It In A Breakthrough Product	bla bla bla...	2012-05-03 00:33:36
2	2	Fox News Anchor Went On An EPIC Rant About The GOP...	bla bla bla...	2012-05-03 00:33:36
3	3	Should Startups Get Special Treatment?	bla bla bla...	2012-05-03 00:33:36
4	4	Executive's Plan To Build A Huge Home Has Enraged ...	bla bla bla...	2012-05-03 00:33:36
5	5	Looking For A New Job Title? This Hilarious Websit...	bla bla bla...	2012-05-03 00:33:36
6	6	Yelp Said Something On Its Earnings Call Today Tha...	bla bla bla...	2012-05-03 00:33:36
7	5	17 Enterprise Startups To Bet Your Career On	bla bla bla...	2012-05-03 00:33:36
8	6	Windows 'Live' Is Dead	bla bla bla...	2012-05-03 00:33:36



id	email	name
1	steve@gmail.com	Steve
2	grace@gmail.com	Grace
3	Alyson@gmail.com	Alyson
4	seth@gmail.com	Seth
5	julie@gmail.com	Julie
6	matt@gmail.com	Matt

Solution: Normalization

- We’ve no removed all redundancy.
- Updating an author’s email address will only happen in one spot, as opposed to every entry they’ve posted.
- Our database will take up much less hard drive space.
- Great, how do we query for this information now?

SQL: Structured Query Language

- Standardized language for communicating with the database.
- Create databases and tables, save, retrieve, update, and delete information.
- SQL is very similar across all RDBMS, however, syntax will vary. Very transferable skill.

SQL: Data Types

- Type and size of each column must be defined.
- Similar to types in PHP (int, boolean, etc...).
- More strict that PHP; DB returns errors if data doesn't match type.
- Categories of types are string, number, and date.

Numeric Type	Description	Range of Values
TINYINT	Very small integer	-128 to 127 0 to 255 if UNSIGNED
SMALLINT	Small integer	-32768 to 32767 0 to 65536 if UNSIGNED
MEDIUMINT	Medium-sized integer	-8388608 to 8388607 0 to 16777215 if UNSIGNED
INT	Normal-sized integer	-2147483648 to 2147483647 0 to 4294967295 if UNSIGNED
BIGINT	Large integer	-9223372036854775808 to 9223372036854775807 0 to 18446744073709551615 if UNSIGNED
FLOAT	Single-precision float	Smallest non-zero: +/-1.176 x 10^-38 Largest value: +/-3.403 x 10^38
DOUBLE	Double-precision float	Smallest non-zero: +/-2.225 x 10^-308 Largest value: +/-1.798 x 10^308
DECIMAL (*precision*, *scale*)	Fixed-point number	Same as Double
BIT	0 or 1	0 or 1

Numeric Data Types

- Choose the type that best suits your needs carefully.
- TINYINT takes 1 byte, SMALLINT takes 2 bytes, MEDIUMINT takes 3 bytes, INT takes 4 bytes, BIGINT takes 8 bytes.
- UNSIGNED means only positive numbers.

Date/Time Type	Description	Range of Values
DATE	Date	1 Jan 1000 to 31 Dec 9999
DATETIME	Date and Time	Midnight, 1 Jan 1000 to 23:59:59, 31 Dec 9999
TIMESTAMP	Timestamp	00:00:01, 1 Jan 1970 to 03:14:07, 9 Jan 2038, UTC
TIME	Time	-838:59:59 to 838:59:59
YEAR	Year	1901 to 2155

Date/Time Types

String Type	Description	Range
CHAR(*n*)	Fixed length string of *n* chars	0-255 chars
VARCHAR(*n*)	Variable-length string of up to *n* chars	0-65535 chars
BINARY(*n*)	Fixed length binary string of *n* bytes	0-255 bytes
VARBINARY(*n*)	Variable length binary string of up to *n* bytes	0-65535 bytes
TINYTEXT	Small Text Field	0-255 chars
TEXT	Normal-sized text field	0-65535 chars
MEDIUMTEXT	Medium-sized text field	0-16777215 chars
LONGTEXT	Large text field	0-4294967295 chars
TINYBLOB	Small BLOB (binary large object)	0-255 bytes
BLOB	Normal-sized BLOB	0-65535 bytes
MEDIUMBLOB	Medium-sized BLOB	0-16777215 bytes
LONGBLOB	Large BLOB	0-4294967295 bytes
ENUM	Enumeration	up to 65,535 values
SET	A set of values	0-64 values

String Types - char and varchar

- name1 char(15) vs name2 varchar(15)
- name1 will always take up 15 characters. 'Dave' will be padded with 11 spaces.
- name2 will only take up as much as you insert. 'Dave' will only be 4 characters
- varchar is not always the best choice though, more overhead for MySQL.

Binary Fields - binary, varbinary, *blob

- Can store binary data in the database.
- Images, Microsoft Word docs, PDFs, etc...
- Not recommended, filesystems are specialized for this.
- Other use-cases as well: Bitwise operations, case sensitive storage.

Types Used in This Class

- int, decimal
- timestamp, datetime
- char, varchar, text, enum

Indexes/Keys

- indexes allow your queries to run much faster.
- indexes can be applied to one column, or a combination of columns.
- Data isn't stored in any particular order, the DB server stores them arbitrarily.
- Without indexes, database must scan entire table, regardless of whether there is only one match or 50.

Indexes/Keys

- An index is separate from the table, and it's stored as a sorted list.
- Ex: SELECT * WHERE `name` = 'Rafael';
- DB knows exactly where to find this in the index, no full table scan is necessary.

Indexes/Keys

- Too many indexes slow down the database because each index has to be updated when a new row is created, or an old row is updated.
- Only index columns that are searched against or sorted by most frequently.

Primary Key Index

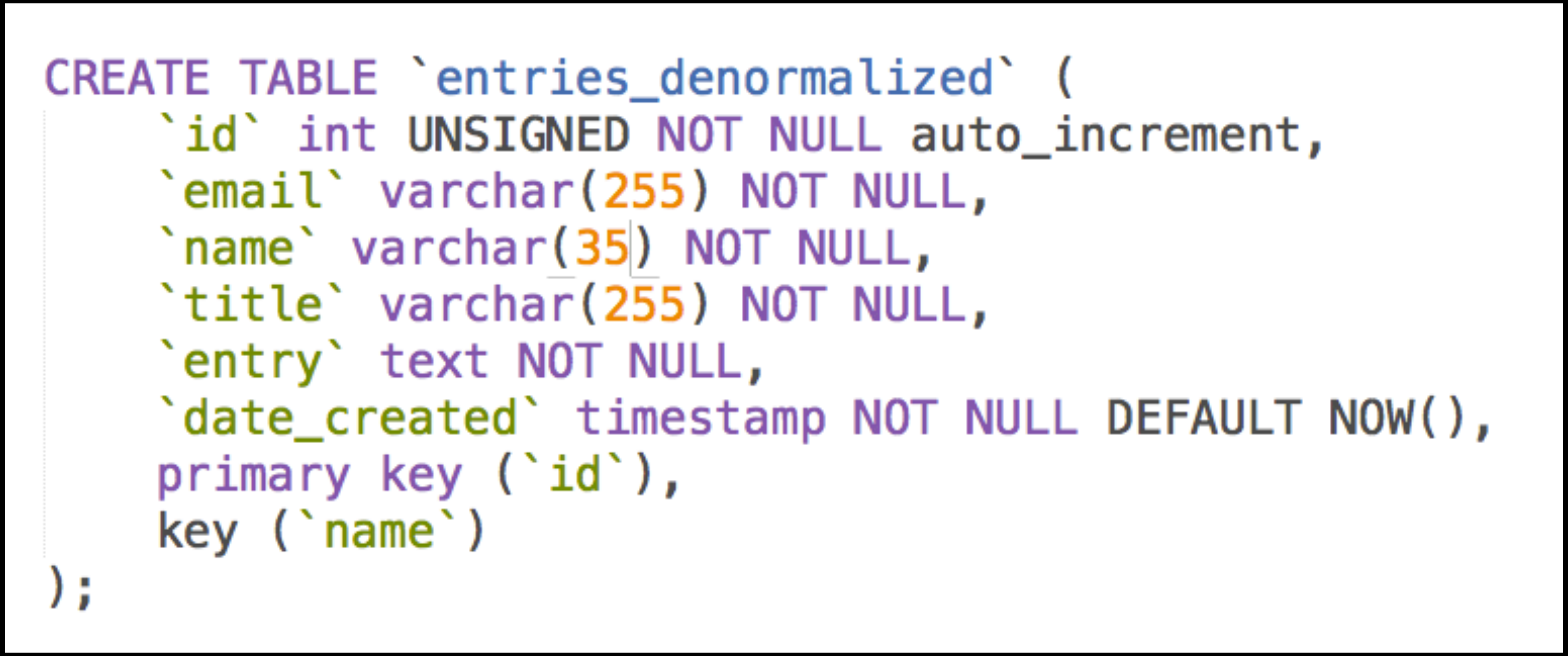
- Every table should have a primary key.
- Every table can have at most one primary key.
- It’s what identifies a row in the table as unique.
- It’s what’s used when relating a row in one table, to a row in another table.

Full Text Search

- MySQL has support for full-text indexing and searching.
- A full-text index in MySQL is an index of type FULLTEXT.
- Full-text indexes can be used only with MyISAM tables, and can be created only for CHAR, VARCHAR, or TEXTcolumns.

SQL - Creating Tables

Back-tics are used here, but not required.
MySQL syntax is forgiving in a lot of ways.



```
CREATE TABLE `entries_denormalized` (
    `id` int UNSIGNED NOT NULL auto_increment,
    `email` varchar(255) NOT NULL,
    `name` varchar(35) NOT NULL,
    `title` varchar(255) NOT NULL,
    `entry` text NOT NULL,
    `date_created` timestamp NOT NULL DEFAULT NOW(),
    primary key (`id`),
    key (`name`)
);
```

SQL - Creating Tables

- id has auto_increment and is set to be the table's primary key.
- Notice UNSIGNED and NOT NULL.
- name is another key (index), can you think of why?
- Can we add another key here that makes sense?
- We can add defaults, see date_created field.

SQL - INSERT INTO
Inserting Data

```
INSERT INTO table_name VALUES (value1, value2);

INSERT INTO table_name ( field1, field2 )
VALUES (value1, value2);
```

- Fields need not be present if values for all columns are specified.
- Values must correspond to the fields.

SQL - SELECT Retrieving Data

- Retrieving all columns for all rows.
- Retrieving all columns for all rows and sort by a column.
- Retrieving specific columns for all rows.
- Retrieving rows based on some criteria.
- Retrieving a subset of rows.

SQL - SELECT

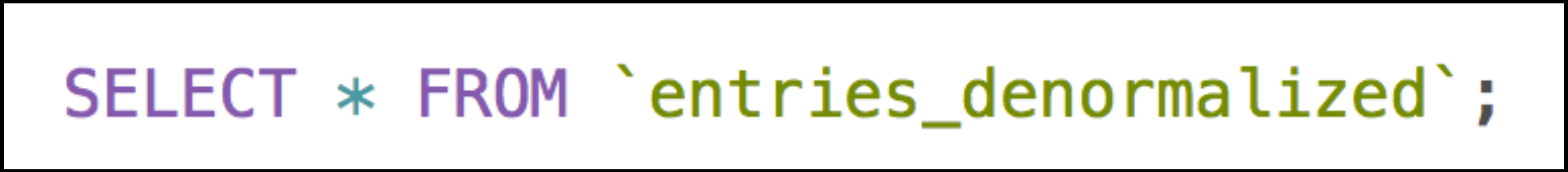
```
SELECT * FROM table_name;

SELECT * FROM table_name
ORDER BY col ASC|DESC;

SELECT col_name1, col_name2 FROM table_name;

SELECT col_name1, col_name2 FROM table_name
WHERE col_name3 = ‘some_value’;

SELECT col_name1, col_name2 FROM table_name
WHERE col_name3 = ‘some_value’ LIMIT 5, 15;
```

SQL - WHERE Clause

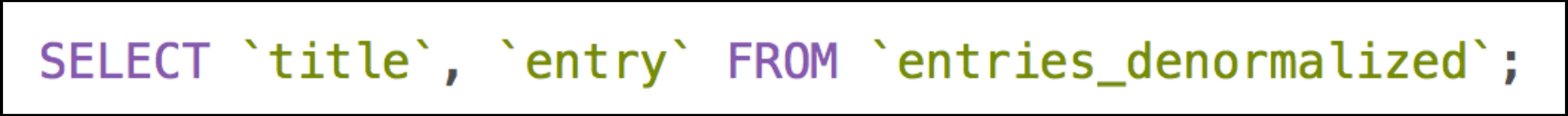
Using AND and OR

```
SELECT col_name1, col_name2 FROM table_name
WHERE col_name3 = ‘some_value’;

SELECT col_name1, col_name2 FROM table_name
WHERE col_name3 = ‘val’ AND col_name4 = ‘val’;

SELECT col_name1, col_name2 FROM table_name
WHERE col_name3 = ‘val’ OR col_name4 = ‘val’;
```

SQL - SELECT Retrieving Data



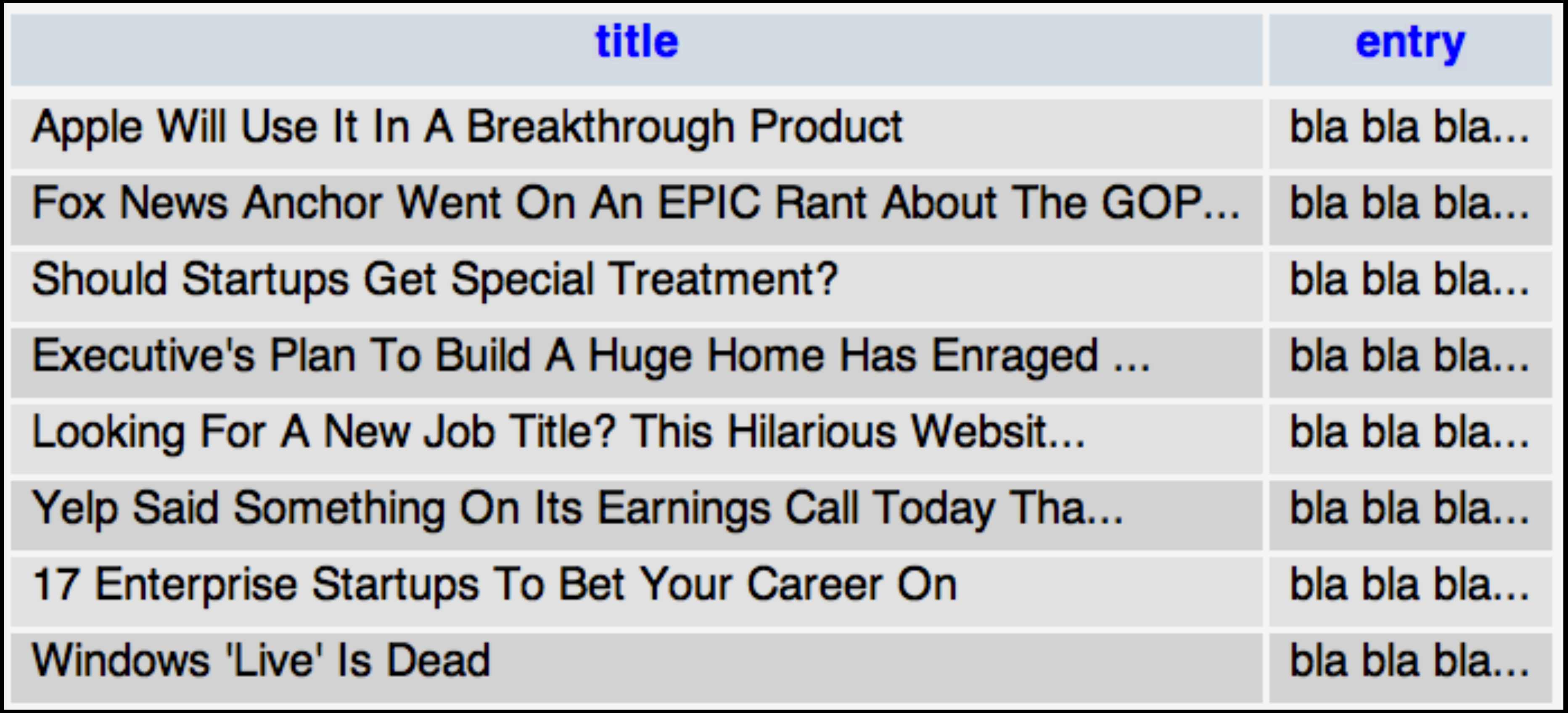
```sql
SELECT * FROM `entries_denormalized`;
```



```sql
SELECT `title`, `entry` FROM `entries_denormalized`;
```

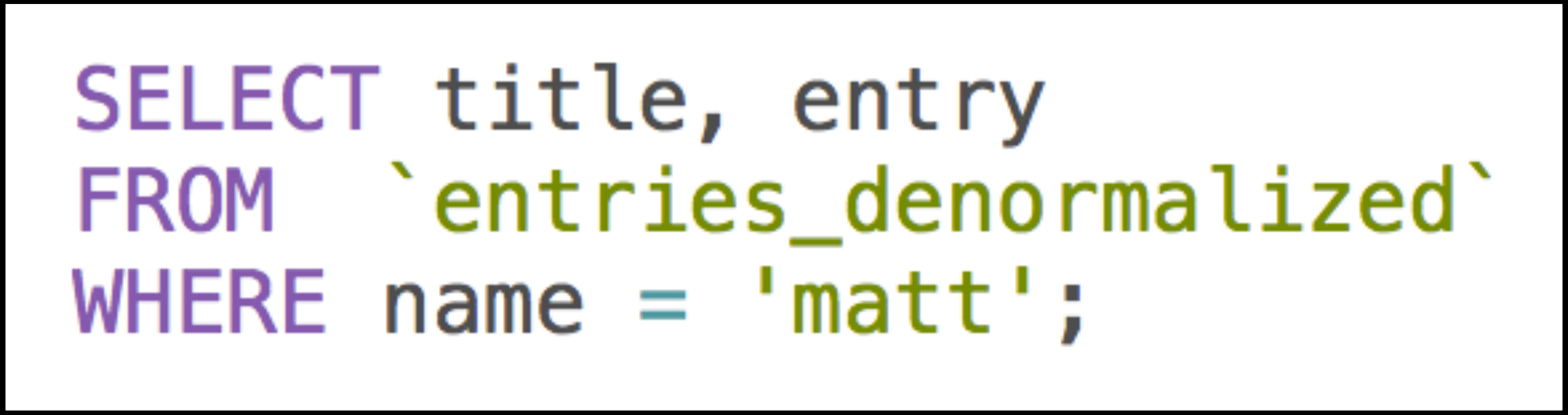
SQL - SELECT
Retrieving Data

```sql
SELECT `title`, `entry` FROM `entries_denormalized`;
```

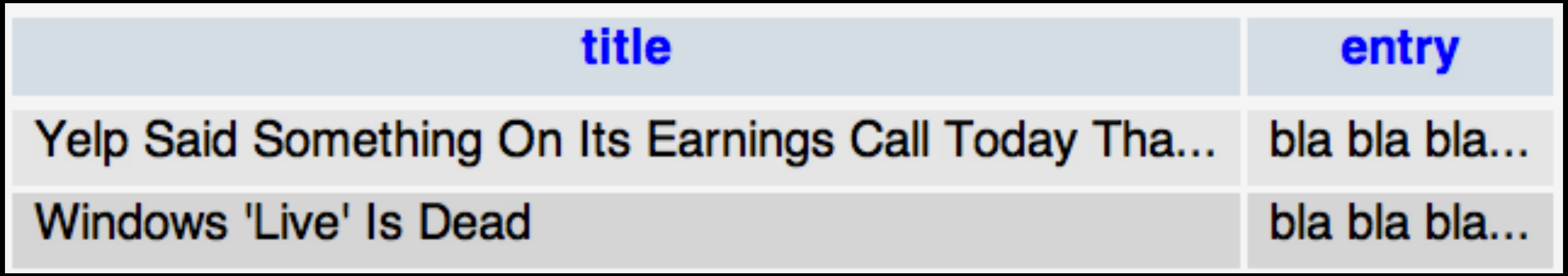


title	entry
Apple Will Use It In A Breakthrough Product	bla bla bla...
Fox News Anchor Went On An EPIC Rant About The GOP...	bla bla bla...
Should Startups Get Special Treatment?	bla bla bla...
Executive's Plan To Build A Huge Home Has Enraged ...	bla bla bla...
Looking For A New Job Title? This Hilarious Websit...	bla bla bla...
Yelp Said Something On Its Earnings Call Today Tha...	bla bla bla...
17 Enterprise Startups To Bet Your Career On	bla bla bla...
Windows 'Live' Is Dead	bla bla bla...

SQL - SELECT
Retrieving Data



```sql
SELECT title, entry
FROM  `entries_denormalized`
WHERE name = 'matt';
```



title	entry
Yelp Said Something On Its Earnings Call Today Tha...	bla bla bla...
Windows 'Live' Is Dead	bla bla bla...

SQL - UPDATE Modifying Rows

UPDATE *table_name* SET *col_name* = ‘new_value’
WHERE *col_name2* = ‘some_value’;

- Columns will be updated in the Rows that match the WHERE clause.
- This can be one or many rows at a time.
- The WHERE portion of the update usually uses a primary key column when updating one row.

SQL - UPDATE
Modifying Rows

```
UPDATE table_name
SET col1 = ‘value’, col2 = ‘value’, col3 = ‘value’
WHERE col4 = ‘value’;
```

- Multiple columns can be updated at one time by separating *col* = ‘value’ by a comma.
- This is more efficient than running multiple UPDATE statements on the rows needing updates.

SQL - DELETE
Removing Rows

```
DELETE FROM table_name
WHERE col = 'value';
```

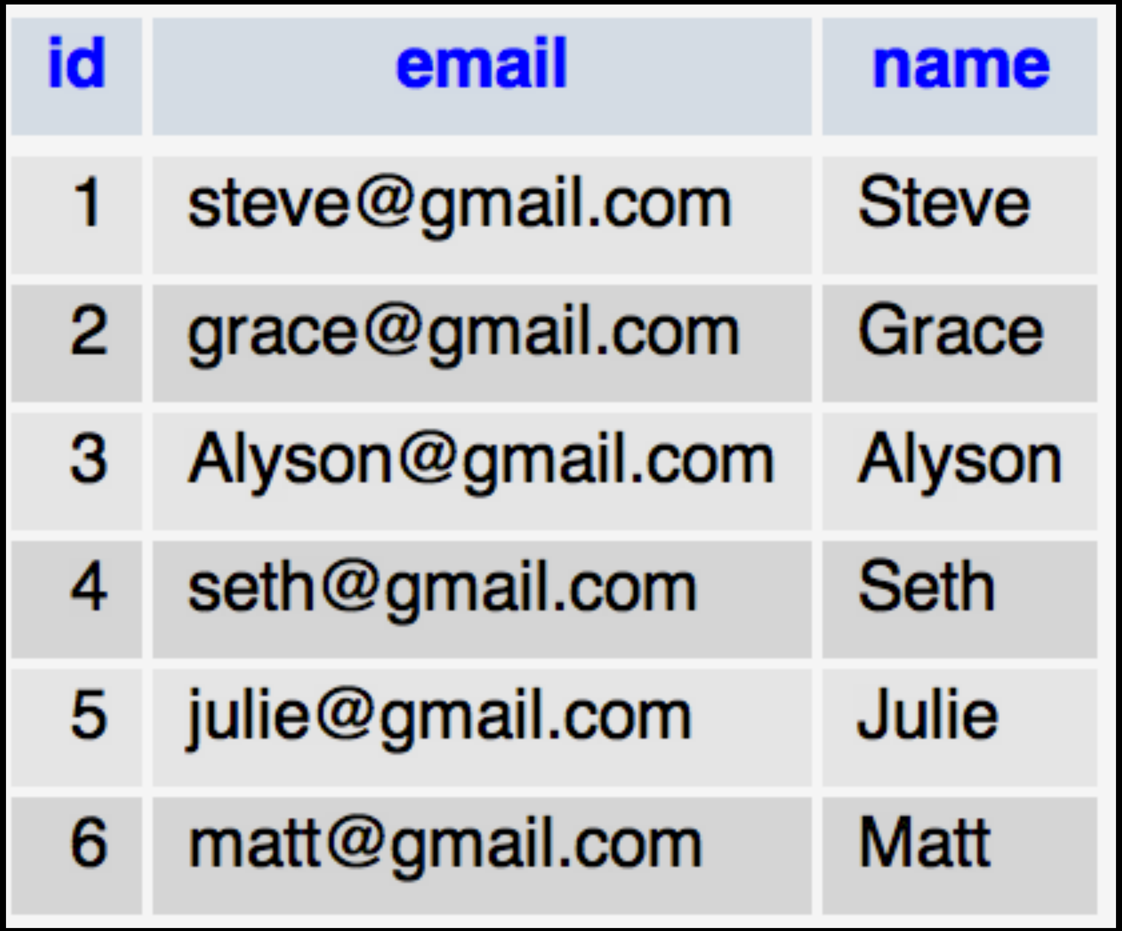
- Rows will be deleted that match the WHERE clause.
- This can be one or many rows at a time.
- The WHERE portion of the update usually uses a primary key column when updating one row.

Back to SELECT INNER JOIN

- This is not in the book and it's important.
- INNER JOIN is used to associated data in one table, with data in another.
- Relational Databases would not be called Relational Databases if this were not possible.
- Back to the example from the beginning of the class, normalization.

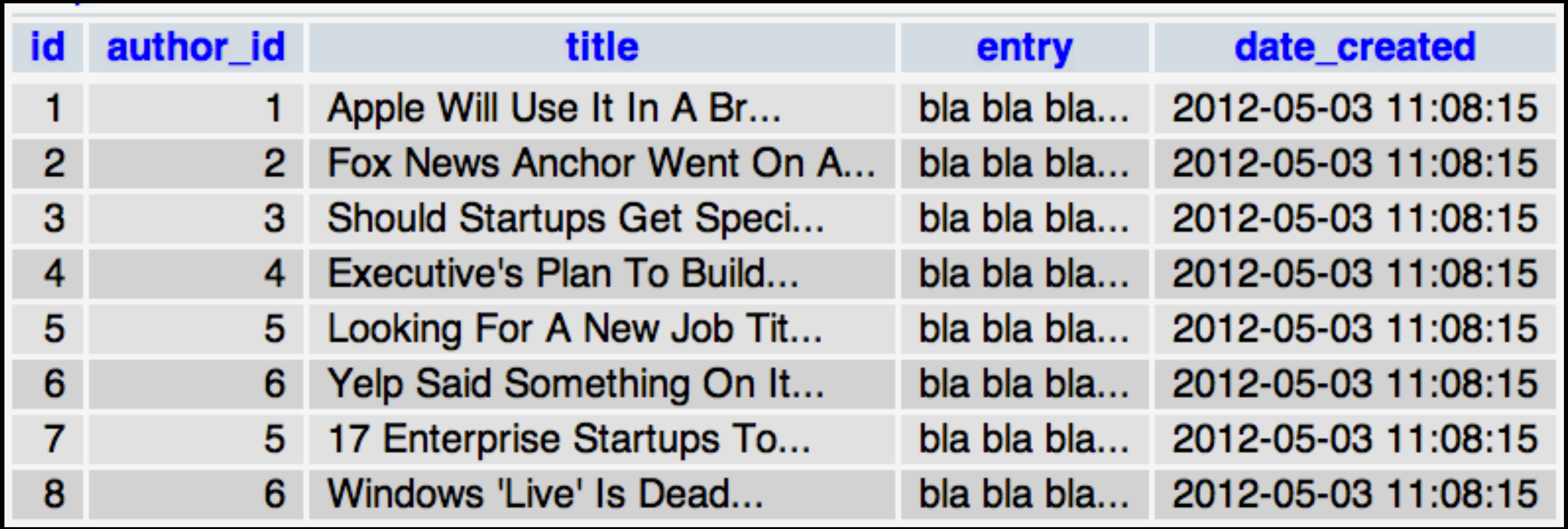
```
SELECT * FROM authors;
SELECT * FROM entries;
```

Authors



id	email	name
1	steve@gmail.com	Steve
2	grace@gmail.com	Grace
3	Alyson@gmail.com	Alyson
4	seth@gmail.com	Seth
5	julie@gmail.com	Julie
6	matt@gmail.com	Matt

Entries



id	author_id	title	entry	date_created
1	1	Apple Will Use It In A Br...	bla bla bla...	2012-05-03 11:08:15
2	2	Fox News Anchor Went On A...	bla bla bla...	2012-05-03 11:08:15
3	3	Should Startups Get Speci...	bla bla bla...	2012-05-03 11:08:15
4	4	Executive's Plan To Build...	bla bla bla...	2012-05-03 11:08:15
5	5	Looking For A New Job Tit...	bla bla bla...	2012-05-03 11:08:15
6	6	Yelp Said Something On It...	bla bla bla...	2012-05-03 11:08:15
7	5	17 Enterprise Startups To...	bla bla bla...	2012-05-03 11:08:15
8	6	Windows 'Live' Is Dead...	bla bla bla...	2012-05-03 11:08:15

INNER JOIN

- We want Entry data as well as Author data.
- We can write two queries and have PHP tie the data together, but that’s really inefficient and impractical.
- We want to use one query to retrieve all of the columns, not multiple queries, and no PHP... This is the solution.

INNER JOIN

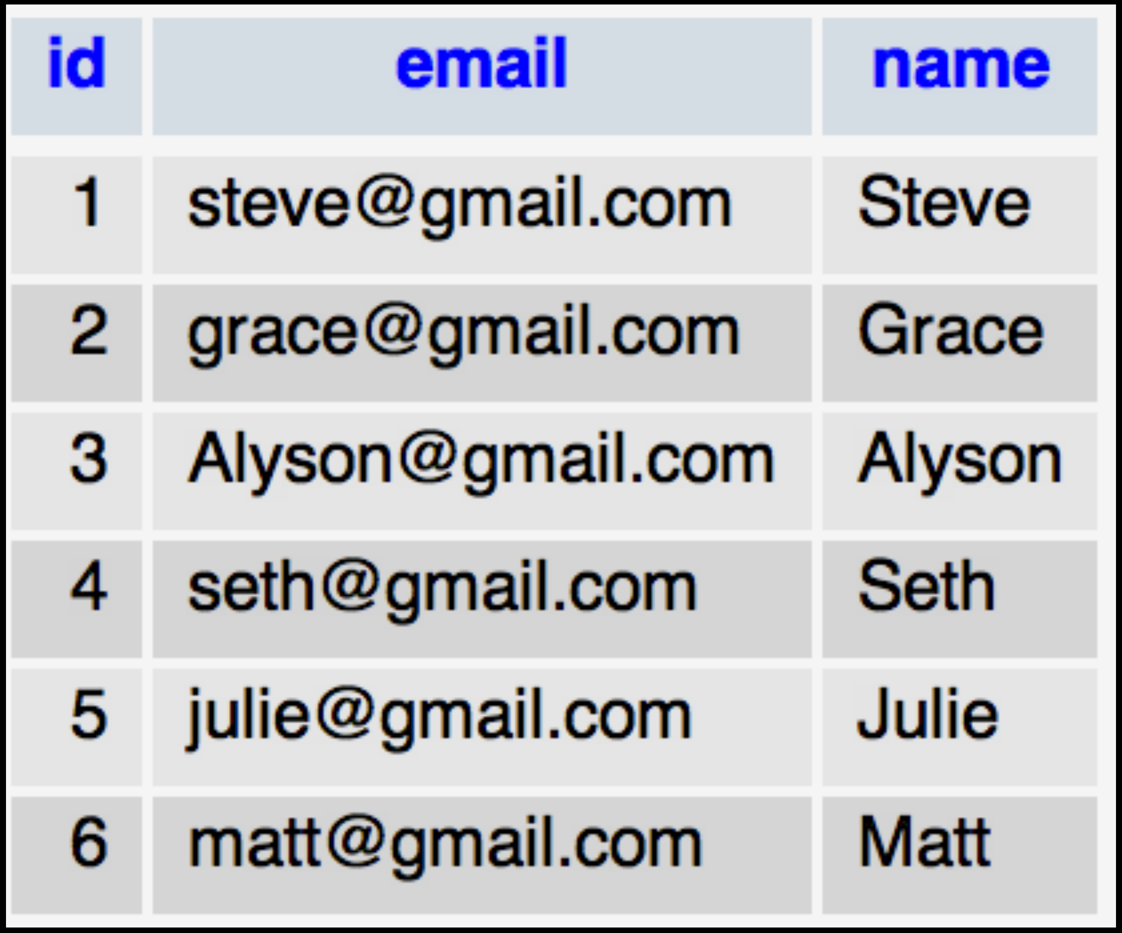
```
SELECT table1.col, table2.col
FROM table1
INNER JOIN table2
    ON table1.col2 = table2.col2
```

ON is the “where” clause that matches a column in one table with a column in the other.

INNER JOIN

- We’re joining two tables together on a common field.
- For example, author id in authors table, and author id in the entries table.
- Will go through all of the rows in the entries table, if it cannot find the author id in the authors table, the row IS NOT returned.

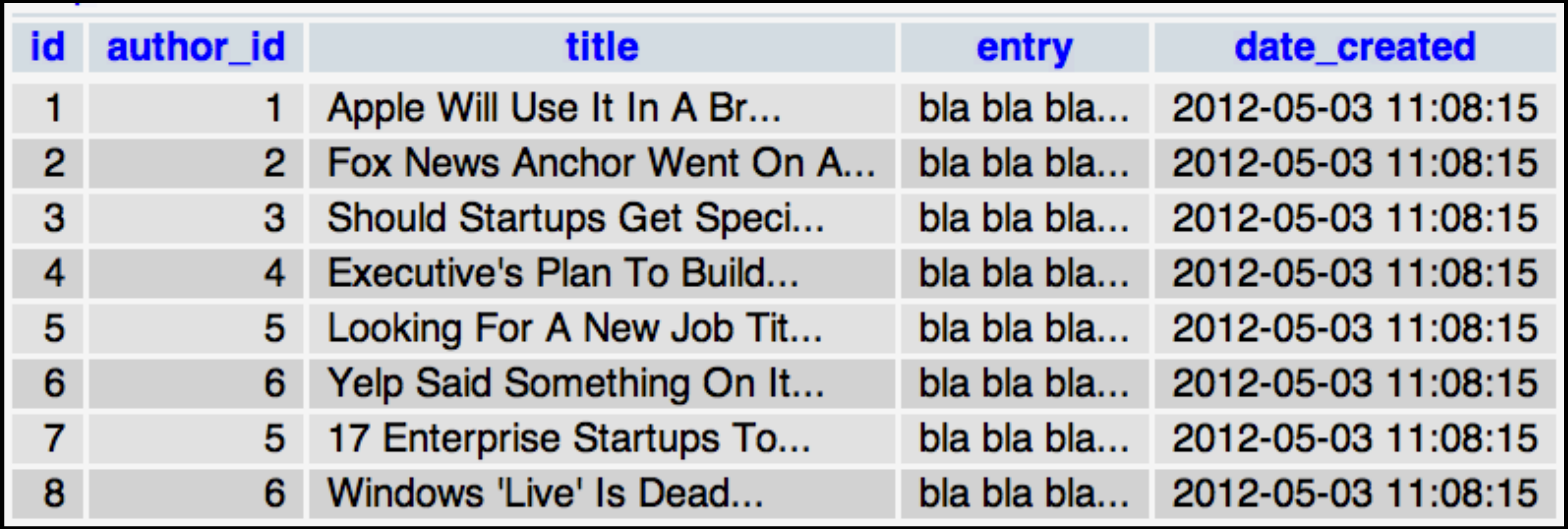
Authors



id	email	name
1	steve@gmail.com	Steve
2	grace@gmail.com	Grace
3	Alyson@gmail.com	Alyson
4	seth@gmail.com	Seth
5	julie@gmail.com	Julie
6	matt@gmail.com	Matt

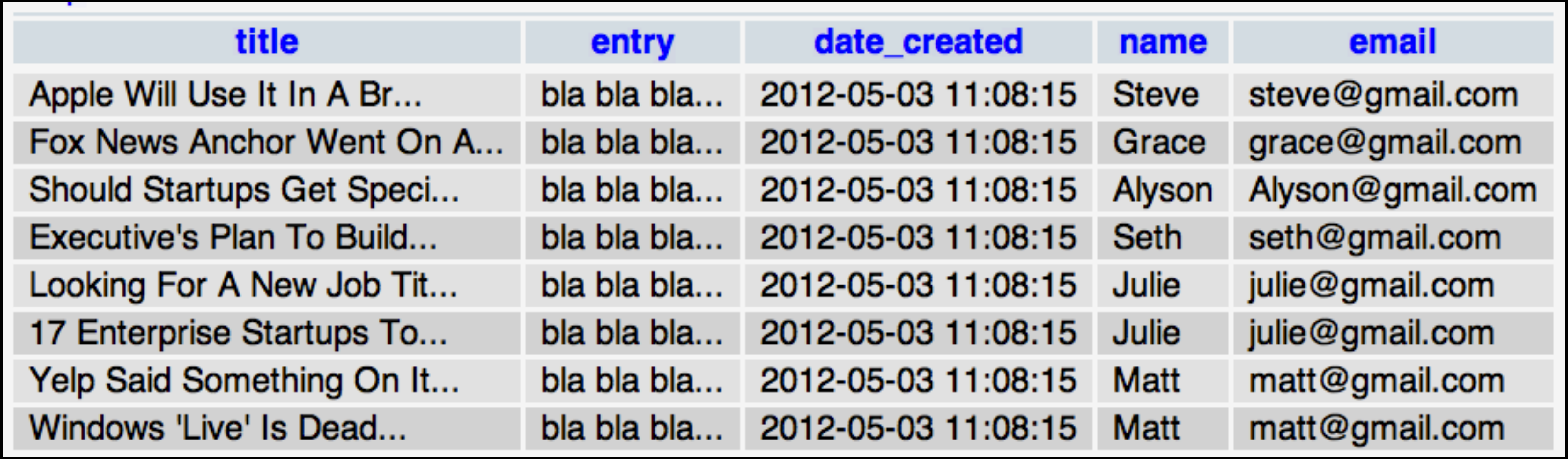
```
SELECT * FROM authors;
SELECT * FROM entries;
```

Entries

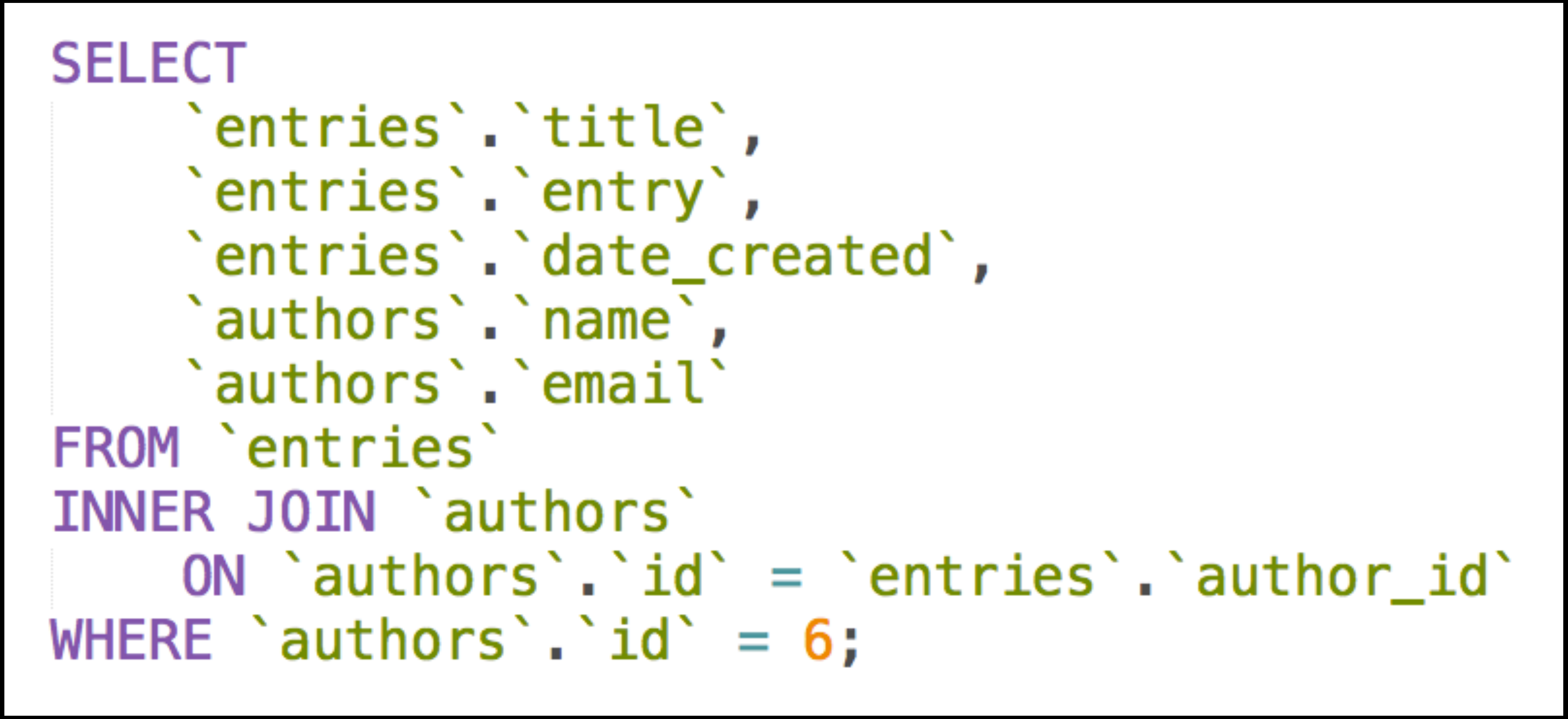


id	author_id	title	entry	date_created
1	1	Apple Will Use It In A Br...	bla bla bla...	2012-05-03 11:08:15
2	2	Fox News Anchor Went On A...	bla bla bla...	2012-05-03 11:08:15
3	3	Should Startups Get Speci...	bla bla bla...	2012-05-03 11:08:15
4	4	Executive's Plan To Build...	bla bla bla...	2012-05-03 11:08:15
5	5	Looking For A New Job Tit...	bla bla bla...	2012-05-03 11:08:15
6	6	Yelp Said Something On It...	bla bla bla...	2012-05-03 11:08:15
7	5	17 Enterprise Startups To...	bla bla bla...	2012-05-03 11:08:15
8	6	Windows 'Live' Is Dead...	bla bla bla...	2012-05-03 11:08:15

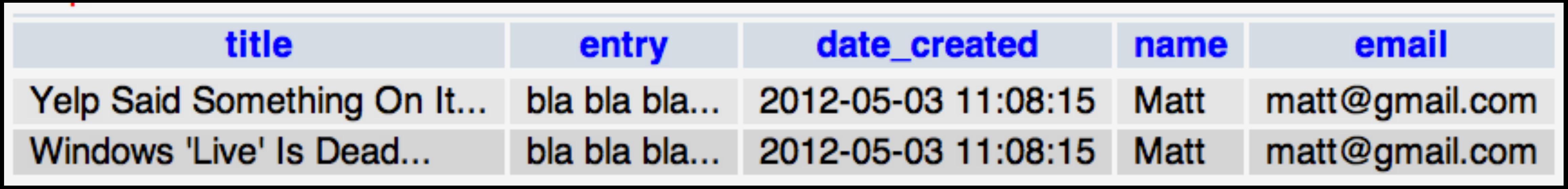
```sql
SELECT
	`entries`.`title`,
	`entries`.`entry`,
	`entries`.`date_created`,
	`authors`.`name`,
	`authors`.`email`
FROM `entries`
INNER JOIN `authors`
	ON `authors`.`id` = `entries`.`author_id`;
```



title	entry	date_created	name	email
Apple Will Use It In A Br...	bla bla bla...	2012-05-03 11:08:15	Steve	steve@gmail.com
Fox News Anchor Went On A...	bla bla bla...	2012-05-03 11:08:15	Grace	grace@gmail.com
Should Startups Get Speci...	bla bla bla...	2012-05-03 11:08:15	Alyson	Alyson@gmail.com
Executive's Plan To Build...	bla bla bla...	2012-05-03 11:08:15	Seth	seth@gmail.com
Looking For A New Job Tit...	bla bla bla...	2012-05-03 11:08:15	Julie	julie@gmail.com
17 Enterprise Startups To...	bla bla bla...	2012-05-03 11:08:15	Julie	julie@gmail.com
Yelp Said Something On It...	bla bla bla...	2012-05-03 11:08:15	Matt	matt@gmail.com
Windows 'Live' Is Dead...	bla bla bla...	2012-05-03 11:08:15	Matt	matt@gmail.com



```sql
SELECT
    `entries`.`title`,
    `entries`.`entry`,
    `entries`.`date_created`,
    `authors`.`name`,
    `authors`.`email`
FROM `entries`
INNER JOIN `authors`
    ON `authors`.`id` = `entries`.`author_id`
WHERE `authors`.`id` = 6;
```



title	entry	date_created	name	email
Yelp Said Something On It...	bla bla bla...	2012-05-03 11:08:15	Matt	matt@gmail.com
Windows 'Live' Is Dead...	bla bla bla...	2012-05-03 11:08:15	Matt	matt@gmail.com

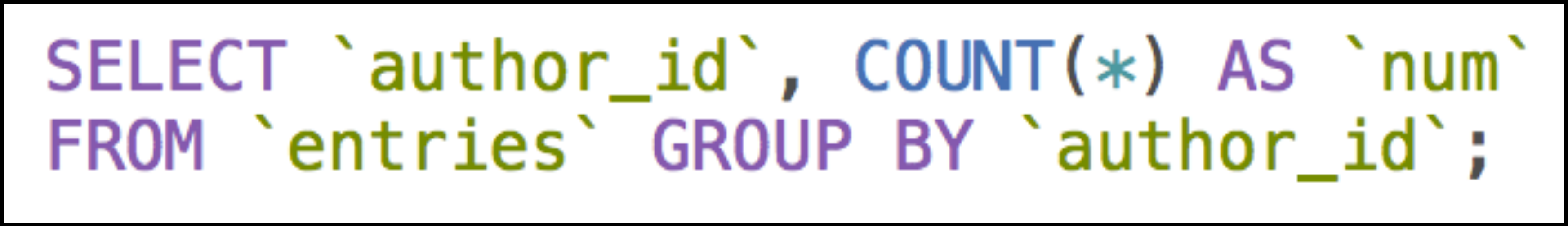
Aggregate Functions - Counting

- This is not in the book.
- We want to know how many entries the author with the ID of 5 has created.
- This requires us to use MySQL function COUNT().
- This requires us to use MySQL aggregator GROUP BY.

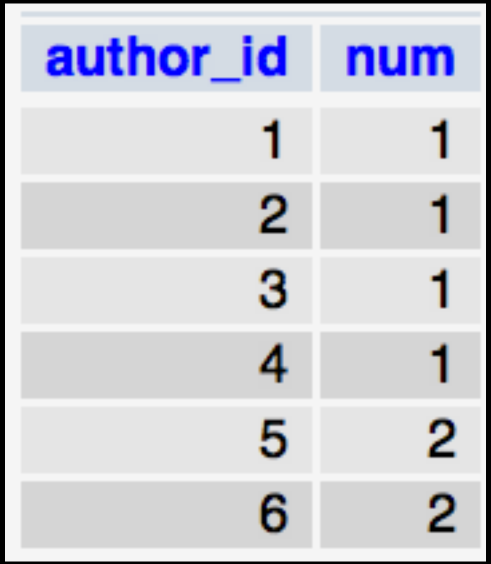
Counting and Grouping E-commerce Examples:

- counting how many products in the table are greater than $4.99.
- counting how many products in the table are in the ‘movie’ category.
- counting how many people purchased a particular item on a specific date.
- etc...

COUNT, GROUP BY



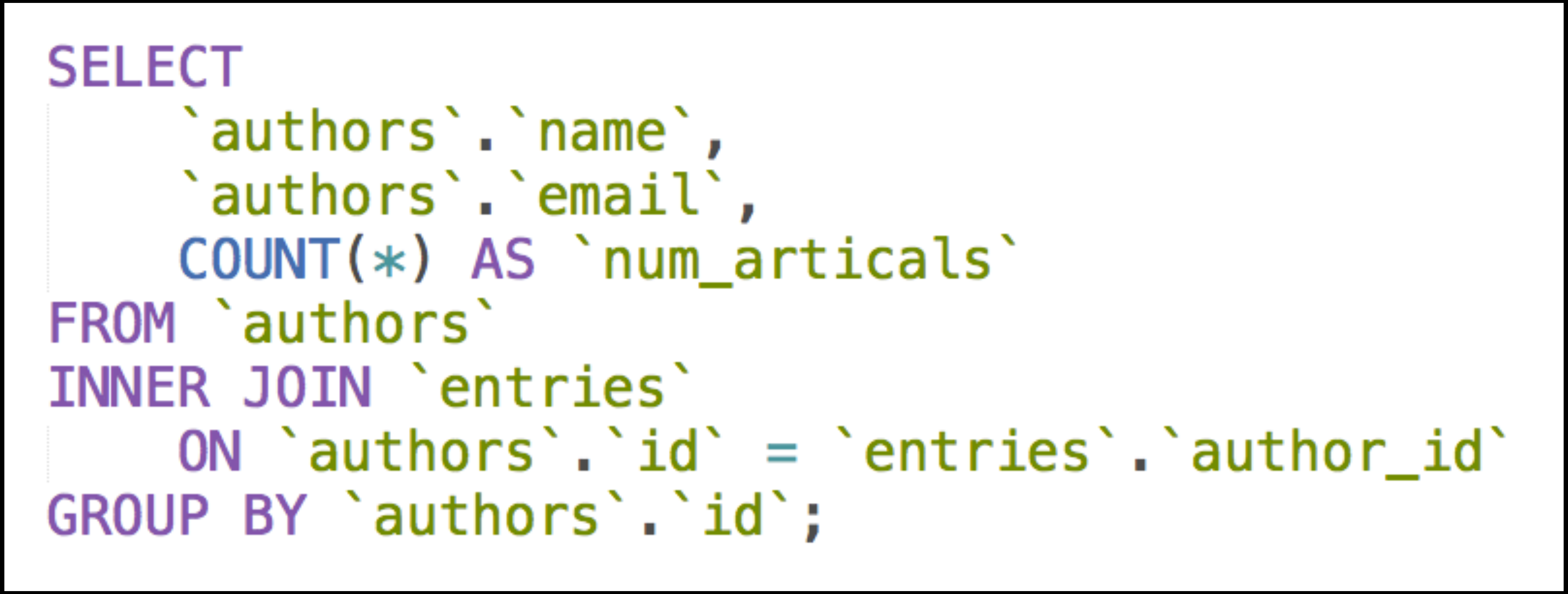
```
SELECT `author_id`, COUNT(*) AS `num`
FROM `entries` GROUP BY `author_id`;
```



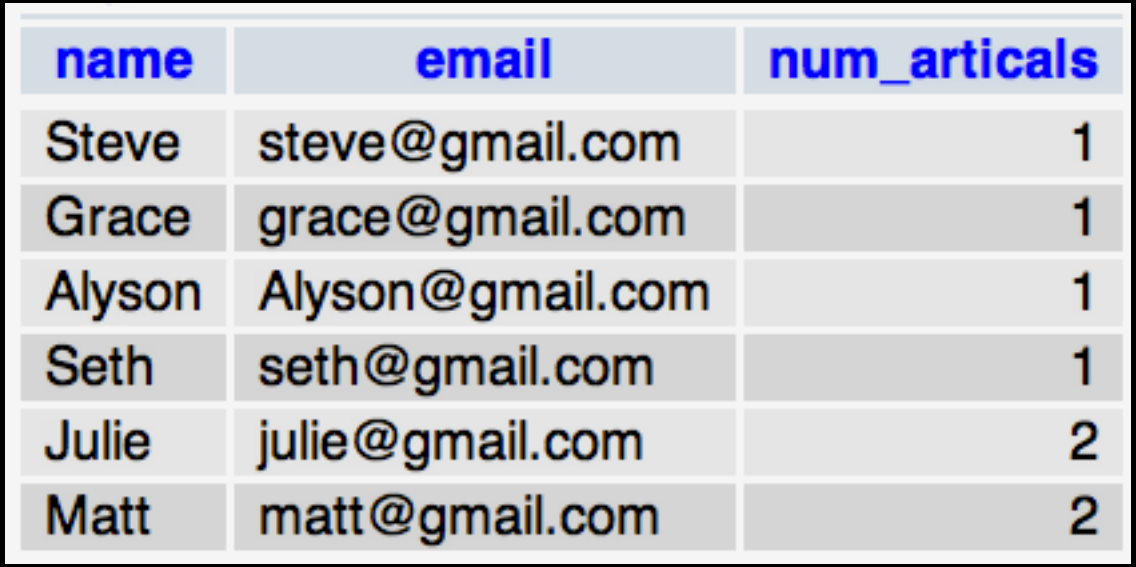
author_id	num
1	1
2	1
3	1
4	1
5	2
6	2

* Notice the alias, using the keyword AS

COUNT, GROUP BY



```
SELECT
    `authors`.`name`,
    `authors`.`email`,
    COUNT(*) AS `num_articals`
FROM `authors`
INNER JOIN `entries`
    ON `authors`.`id` = `entries`.`author_id`
GROUP BY `authors`.`id`;
```



name	email	num_articals
Steve	steve@gmail.com	1
Grace	grace@gmail.com	1
Alyson	Alyson@gmail.com	1
Seth	seth@gmail.com	1
Julie	julie@gmail.com	2
Matt	matt@gmail.com	2