

SQL Syntax

Housekeeping

- Don't forget to sign in!

Whole Database Commands

```
CREATE {DATABASE | SCHEMA} [IF NOT  
EXISTS] db_name
```

```
ALTER {DATABASE | SCHEMA} [db_name]  
alter_specification ...
```

```
DROP {DATABASE | SCHEMA} [IF EXISTS]  
db_name
```

Table Commands

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS]  
tbl_name
```

```
ALTER [ONLINE | OFFLINE] [IGNORE] TABLE  
tbl_name
```

```
DROP [TEMPORARY] TABLE [IF EXISTS]  
tbl_name
```

```
TRUNCATE [TABLE] tbl_name
```

CREATE Examples

```
CREATE TABLE t1 (a INTEGER, b CHAR(10));
```


ALTER TABLE Examples

```
ALTER TABLE t1 RENAME t2;
```

```
ALTER TABLE t2 ADD d TIMESTAMP;
```

```
ALTER TABLE t2 ADD INDEX (d), ADD  
UNIQUE (a);
```

DROP and TRUNCATE

```
DROP [TEMPORARY] TABLE [IF EXISTS]  
tbl_name
```

```
TRUNCATE [TABLE] tbl_name
```

(both require DROP Privileges)

SELECT (the basics)

SELECT

FROM

WHERE

ORDER BY

SELECT (MySQL)

SELECT

```
[ALL | DISTINCT | DISTINCTROW ]  
  [HIGH_PRIORITY]  
  [STRAIGHT_JOIN]  
  [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]  
  [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]  
select_expr [, select_expr ...]  
[FROM table_references  
[WHERE where_condition]  
[GROUP BY {col_name | expr | position}  
  [ASC | DESC], ... [WITH ROLLUP]]  
[HAVING where_condition]  
[ORDER BY {col_name | expr | position}  
  [ASC | DESC], ...]  
[LIMIT {[offset,] row_count | row_count OFFSET offset}]  
[PROCEDURE procedure_name(argument_list)]  
[INTO OUTFILE 'file_name' export_options  
  | INTO DUMPFILE 'file_name'  
  | INTO var_name [, var_name]]  
[FOR UPDATE | LOCK IN SHARE MODE]]
```

SELECT Examples

```
SELECT 1;
```

```
SELECT * FROM wp_users;
```

```
SELECT user_login, user_email FROM wp_users;
```

```
SELECT *  
FROM wp_users wp JOIN wp_79_posts wpp  
ON wp.ID = wpp.post_author  
WHERE wpp.post_status='publish'  
ORDER BY wpp.post_date;
```

BEING A
**WEB
DESIGNER**

— IS EASY. IT'S LIKE —

RIDING A BIKE

EXCEPT THE BIKE IS ON FIRE
YOU'RE ON FIRE
EVERYTHING IS ON FIRE
AND YOU'RE IN HELL

Create Movies

- `CREATE TABLE `movies` (`
- ``ID` int(11) NOT NULL AUTO_INCREMENT,`
- ``MovieName` varchar(45) DEFAULT NULL,`
- ``Rating` double DEFAULT NULL,`
- ``DateAdded` timestamp NULL DEFAULT CURRENT_TIMESTAMP,`
- `PRIMARY KEY (`ID`)`
- `) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;`

CREATE Ratings

- `CREATE TABLE `Rating` (`
- ``ID` int(11) NOT NULL AUTO_INCREMENT,`
- ``Rating` int(11) DEFAULT NULL,`
- ``MovieID` int(11) DEFAULT NULL,`
- `PRIMARY KEY (`ID`),`
- `KEY `fk_Rating_Movie_idx` (`MovieID`),`
- `CONSTRAINT `fk_Rating_Movie` FOREIGN KEY (`MovieID`) REFERENCES`
``movies` (`ID`) ON DELETE NO ACTION ON UPDATE NO ACTION`
- `) ENGINE=InnoDB AUTO_INCREMENT=1000001 DEFAULT CHARSET=latin1;`

INSERT Statements

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

INSERT statements

- `INSERT INTO movies (MovieName,
Rating)
VALUES ('Star Wars', 8.0);`

UPDATE statements

- UPDATE *table*
- SET *field = value*
- WHERE *condition*;

UPDATE examples

- UPDATE movies
SET rating = 2
WHERE ID = 1;
- UPDATE movies
SET MovieName = 'Star Wars IV: A New Hope'
WHERE MovieName = 'Star Wars';

DELETE Statements

DELETE FROM *table*

WHERE *condition*;

DELETE Examples

-- Don't do this, deletes everything!

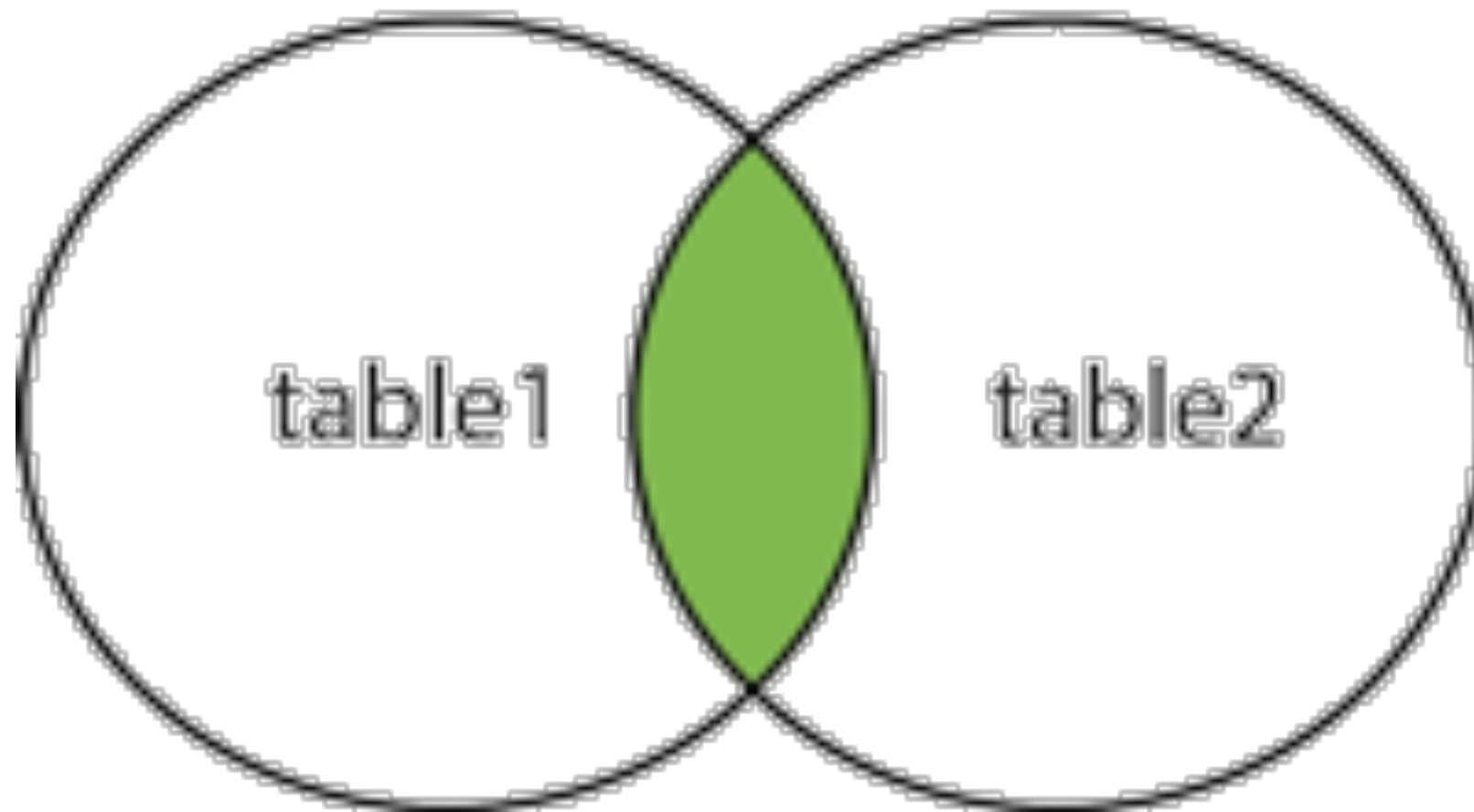
```
DELETE FROM Movies;
```

```
DELETE FROM Movies
```

```
WHERE MovieName LIKE '%Matrix%';
```

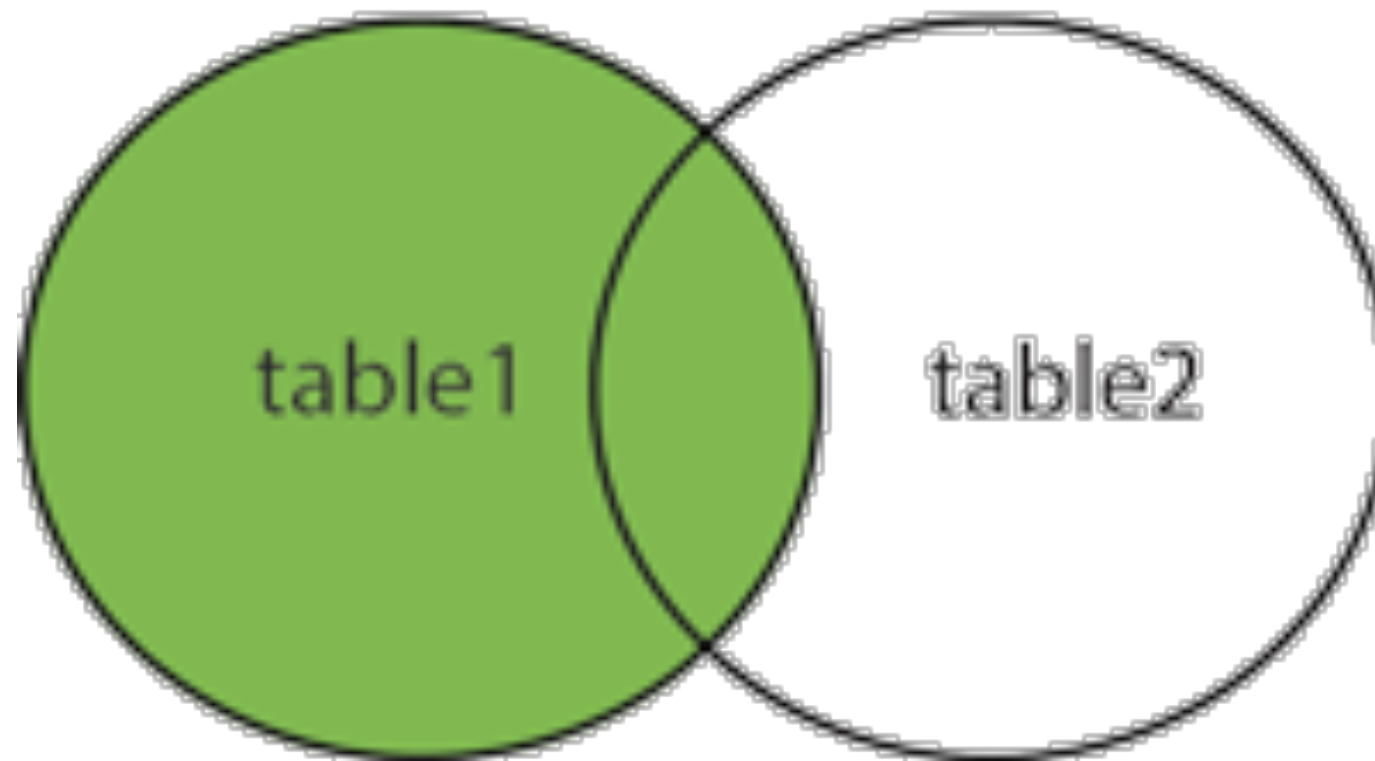
INNER JOIN

INNER JOIN



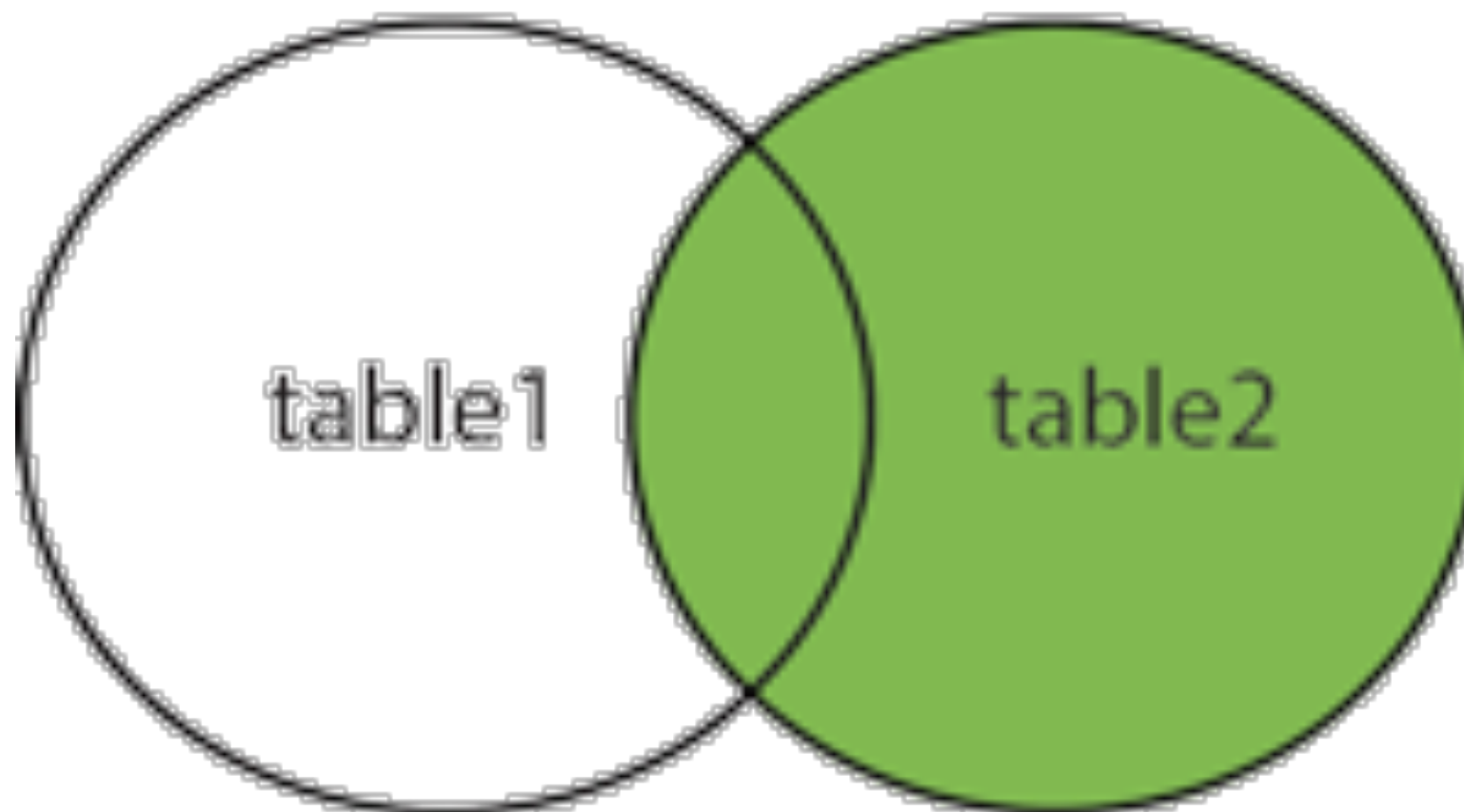
LEFT JOIN

LEFT JOIN



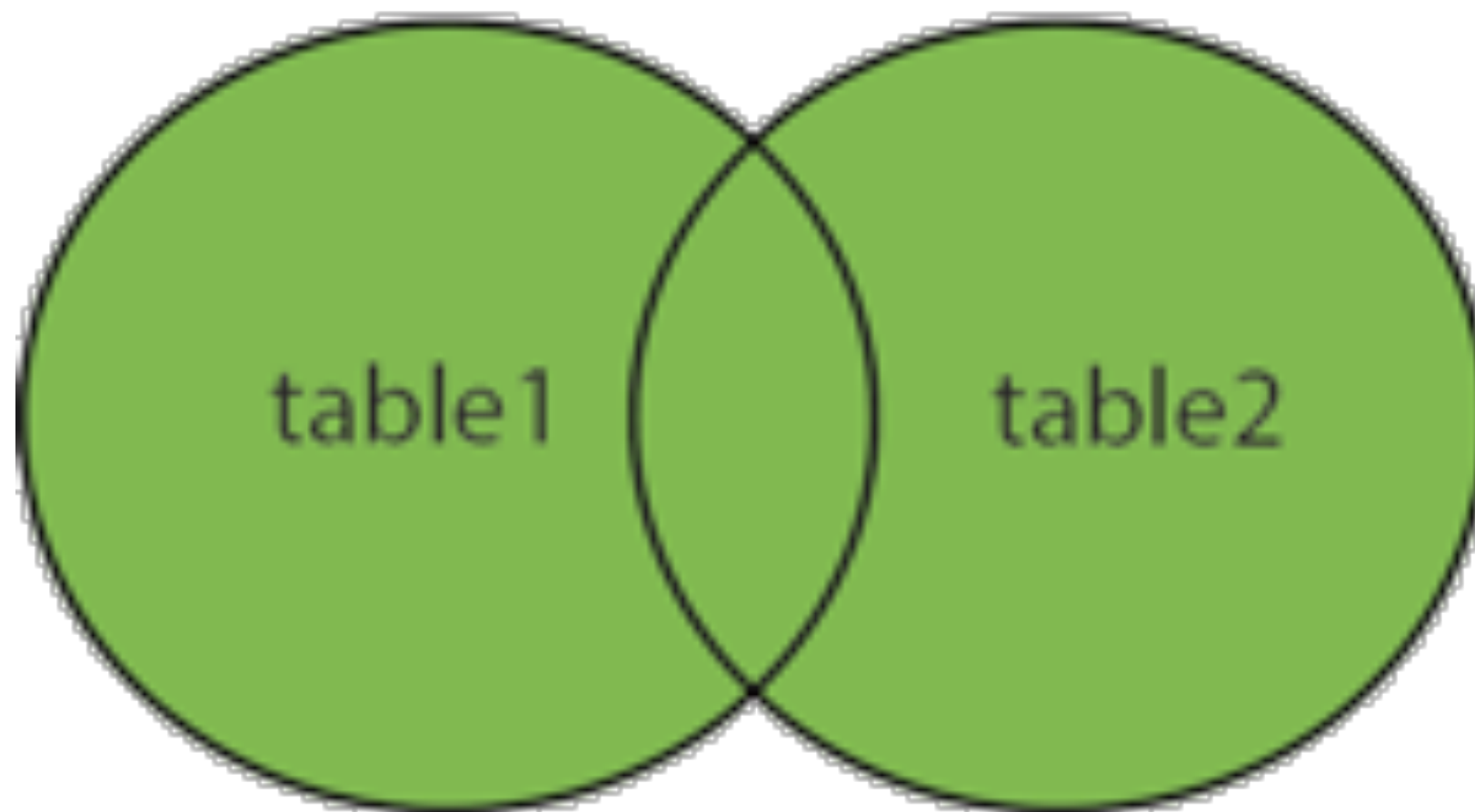
RIGHT JOIN

RIGHT JOIN



FULL OUTER JOIN

FULL OUTER JOIN



INNER JOIN

```
select * from Beer JOIN Batches ON  
Beer.ID = Batches.Beer;
```

INNER JOIN

```
SELECT * FROM Beer be JOIN  
BeerIngredients bi ON be.ID =  
bi.BeerJOIN  
Ingredients i ON bi.Ingredient =  
i.ID;
```

LEFT JOIN

```
SELECT * FROM Beer be JOIN  
BeerIngredients bi ON be.ID =  
bi.BeerJOIN  
Ingredients i ON bi.Ingredient =  
i.ID;
```

FULL-ish OUTER-ish

```
SELECT
```

```
    *
```

```
FROM
```

```
    Beer be
```

```
        LEFT OUTER JOIN
```

```
        BeerIngredients bi ON be.ID = bi.Beer
```

```
        RIGHT OUTER JOIN
```

```
        Ingredients i ON bi.Ingredient = i.ID;
```

```
--Try replacing the RIGHT with LEFT
```

UNION

```
SELECT Name FROM Beer
```

```
UNION
```

```
SELECT Name FROM Ingredients;
```


Stored Procedure

```
DELIMITER //
```

```
CREATE PROCEDURE `add_ingredient` (type int, cost int, name  
varchar(45), current_inventory int, out id int)
```

```
BEGIN
```

```
    INSERT INTO Ingredients (Type, Cost, Name,  
CurrentInventory_Cups) VALUES (type, cost, name,  
current_inventory);
```

```
    SELECT LAST_INSERT_ID( ) INTO id;
```

```
END //
```

Calling Stored Procedures

```
CALL `add_ingredient` (2, 4, 'new ingredient', 20,  
@outgoingid);
```

```
select @outgoingid;
```

FUNCTIONS

```
DELIMITER //
```

```
CREATE FUNCTION `add_ingredient_function` (type int, cost int, name varchar(45),  
current_inventory int)
```

```
RETURNS INT
```

```
BEGIN
```

```
    INSERT INTO Ingredients (Type, Cost, Name, CurrentInventory_Cups) VALUES (type,  
cost, name, current_inventory);
```

```
    RETURN LAST_INSERT_ID();
```

```
END //
```

Calling Functions

```
SELECT `add_ingredient_function` (2, 4, 'new  
ingredient', 20);
```

TRIGGERS

DELIMITER |

```
CREATE TRIGGER trig_average AFTER INSERT ON Rating
```

```
FOR EACH ROW
```

```
BEGIN
```

```
UPDATE movies SET Rating=(SELECT AVG(Rating) FROM Rating  
WHERE MovieID = NEW.MovieID) WHERE ID=NEW.MovieID;
```

```
END;
```

```
|
```

```
INSERT INTO Rating (Rating, MovieID) VALUES (100, 2);
```


Questions from Last Night?

GROUP BY

- Aggregate functions sometimes need a GROUP BY to display results the way you would like.

```
SELECT
```

```
    Beer.Name,
```

```
    SUM(BeerIngredients.Amount_Cups * Ingredients.Cost / 2)
```

```
FROM
```

```
    BeerIngredients
```

```
        JOIN
```

```
        Ingredients ON BeerIngredients.Ingredient = Ingredients.Id
```

```
        JOIN
```

```
        Beer ON BeerIngredients.Beer = Beer.ID
```

```
GROUP BY Beer.Name
```

VIEWS

```
CREATE VIEW beer_costs AS
```

```
SELECT
```

```
    Beer.Name,
```

```
    SUM((BeerIngredients.Amount_Cups) * Ingredients.Cost / 2)
```

```
FROM
```

```
    BeerIngredients
```

```
        JOIN
```

```
        Ingredients ON BeerIngredients.Ingredient = Ingredients.Id
```

```
        JOIN
```

```
        Beer ON BeerIngredients.Beer = Beer.ID
```

```
GROUP BY Beer.Name
```

VIEWS...why?

- Hide Complexity
- Security Mechanism
- Simplify Supporting Legacy Code (if you have to modify underlying tables, but want existing code to keep working)

Advanced SQL Topics - NoSQL

- Very fast write performance
 - Facebook stores 135 billion messages/month
 - Twitter stores 7TB of data per day
- Easily scale
 - Relational Databases generally grow by making a machine more powerful, or sharding which is complicated
 - NoSQL can easily add a server and continue

But first...JSON

- Similar to XML in that it's a method of storing and transferring data

```
<xml>
  <restaurants>
    <restaurant>
      <name>Taco Bell</name>
      <phone>232-323-2323</phone>
    </restaurant>
    <restaurant>
      <name>The Kitchen</name>
      <phone>343-343-3434</phone>
    </restaurant>
  </restaurants>
</xml>
```

JSON version

```
{  
  "restaurants": {  
    "restaurant": [  
      {  
        "name": "Taco Bell",  
        "phone": "232-323-2323"  
      },  
      {  
        "name": "The Kitchen",  
        "phone": "343-343-3434"  
      }  
    ]  
  }  
}
```

Vagrant

- <https://www.vagrantup.com/>

Let's Install MongoDB

Using Mongo in PHP

```
// connect

$m = new MongoClient();

// select a database

$db = $m->comedy;

// select a collection (analogous to a relational database's table)

$collection = $db->cartoons;

// add a record

$document = array( "title" => "Calvin and Hobbes", "author" => "Bill Watterson" );

$collection->insert($document);

// add another record, with a different "shape"

$document = array( "title" => "XKCD", "online" => true );

$collection->insert($document);
```


More MongoDB PHP

```
// find everything in the collection
```

```
$cursor = $collection->find();
```

```
// iterate through the results
```

```
foreach ($cursor as $document) {
```

```
    echo $document["title"] . "\n";
```

```
}
```