Experts say the first 150-year-old has already been born.

If you were 150 years old today, here are some of the iconic headlines you would have seen.

See the front cover on the day you were born.
Administrivia

- Midterm - End of Class
- Homework
  - Homework 6 due 4/4/2013
On The Menu

- GROUP BY
- Aggregate Functions
- Join Reviews
- Reflexive Joins
- Update Syntax
- Delete Syntax
- Complex News Queries
GROUP BY

- A SELECT against a table generates a “virtual” table containing the matching rows.
- Adding a GROUP BY [col_name] statement collapses this virtual table so that there is just one row for each individual value of [col_name].
- GROUP BY statements are often combined with aggregate functions (SUM, MIN, MAX, AVG).
Por Exemplo

**track**
- track_id INT
- artist_name VARCHAR(255)
- album_name VARCHAR(255)
- release_year INT
- track_name VARCHAR(255)
- disc_number INT
- track_number INT
- track_length FLOAT
- track_popularity FLOAT

**Indexes**
CREATE TABLE IF NOT EXISTS `track` (  
  `track_id` INT NOT NULL AUTO_INCREMENT ,  
  `artist_name` VARCHAR(255) NOT NULL ,  
  `album_name` VARCHAR(255) NOT NULL ,  
  `release_year` INT NOT NULL ,  
  `track_name` VARCHAR(255) NULL ,  
  `disc_number` INT NULL ,  
  `track_number` INT NULL ,  
  `track_length` FLOAT NULL ,  
  `track_popularity` FLOAT NULL ,  
  PRIMARY KEY (`track_id`) )  
ENGINE = InnoDB;
LOAD DATA LOCAL INFILE "/Users/evansandhaus/Documents/NYU_DB/class_17/green_day.txt"
INTO TABLE track
FIELDS TERMINATED BY '\t'
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
(
    artist_name,
    album_name,
    release_year,
    track_name,
    disc_number,
    track_number,
    track_length,
    track_popularity
);

Tuesday, April 2, 13
Aggregate Queries

SELECT
    SUM(numeric_column_1),
    MIN(numeric_column_2),
    MAX(numeric_column_3),
    AVG(numeric_column_4),
    COUNT(any_column),
    COUNT(DISTINCT any_column),
FROM
    table_1
    ...

Examples

- Longest Track
- Shortest Track
- Number of uniquely named tracks.
- Average Popularity Per Album.
- Number of tracks released per year.
- Number of tracks longer than average track.
- Most popular track per album.
World’s Simplest Database

```
professor
- professor_id INT(11)
- first_name VARCHAR(255)
- last_name VARCHAR(255)

Indexes

class
- class_id INT(11)
- class_name VARCHAR(255)
- professor_id INT(11)

Indexes
```
JOIN

- SELECT * FROM professor;
- SELECT * FROM class;
- SELECT * FROM professor JOIN class;
  - Returns ALL POSSIBLE combinations of rows in the professor and class tables.
  - If we only want to see the combination of rows that reflect which professor is teaching which class, we must specify some type of join.
INNER JOIN

- When SELECTING an inner join between tables \( t1 \) and \( t2 \) on key \( k \) the only rows that are returned are those rows in which \( t1.k = t2.k \).
  - SELECT * FROM professor JOIN CLASS ON professor.professor_id = class.professor_id;
INNER JOIN - Visualized

INNER JOIN

SELECT *
FROM Table1 t1
INNER JOIN Table2 t2
ON t1.Col1 = t2.Col1

(C) http://blog.SQLAuthority.com
INNER JOIN

- Lots of different ways of expressing - the following are equivalent in MySQL
  - SELECT * FROM t1 INNER JOIN t2 ON t1.k = t2.k;
  - SELECT * FROM t1 JOIN t2 ON t1.k = t2.k;
  - SELECT * FROM t1, t2 WHERE t1.k = t2.k;
OUTER JOIN

- Outer joins come in three flavors.
  - LEFT
  - RIGHT
  - FULL
When SELECTING a LEFT OUTER JOIN from tables t1 and t2 on key k the result contains:

• All rows from t1
• All rows from t2 that match t1 on k or NULL if no match exists.

SELECT * FROM t1 LEFT OUTER JOIN t2 ON t1.k=t2.k;
LEFT OUTER JOIN

```
SELECT *
FROM Table1 t1
LEFT OUTER JOIN Table2 t2
ON t1.Col1 = t2.Col1
```

(C) http://blog.SQLAuthority.com
RIGHT OUTER JOIN

- When SELECTING a RIGHT OUTER JOIN from tables t1 and t2 on key k the result contains:
  - All rows from t2
  - All rows from t1 that match t2 on k or NULL if no match exists.

- SELECT * FROM t1 RIGHT OUTER JOIN t2 ON t1.k=t2.k;
RIGHT OUTER JOIN

```
SELECT *
FROM Table1 t1
RIGHT OUTER JOIN Table2 t2
ON t1.Col1 = t2.Col1
```
FULL OUTER JOIN

When SELECTING a FULL OUTER JOIN from tables t1 and t2 on key k the result contains:

• All rows from t2
• All rows from t1
• NULL values for t1 and t2 rows where there exists no match on key k.

No straightforward way of typing this in MySQL
• will cover in next lecture
FULL OUTER JOIN

SELECT *
FROM Table1 t1
FULL OUTER JOIN Table2 t2
ON t1.Col1 = t2.Col1

(C) http://blog.SQLAuthority.com
OUTER JOIN - Syntax

- The following are equivalent
  - SELECT * FROM t1 LEFT OUTER JOIN t2 ON t1.k = t2.k
  - SELECT * FROM t1 LEFT JOIN t2 ON t1.k = t2.k
UNION

- UNION allows you to combine the results of two selects.

Syntax:

\[(\text{SELECT col}_x,\text{col}_y \text{ FROM table}_1)\]
\[\text{UNION}\]
\[(\text{SELECT col}_z,\text{col}_q \text{ FROM table}_2)\]

UNION FACTS!:

- UNION gives the resulting table col names from t1.
- UNION eliminates duplicate rows by default.
- UNION will fail if the joined columns have incompatible types.
UNION & FULL OUTER JOIN

- (SELECT * FROM t1 RIGHT OUTER JOIN ON t1.k = t2.k)

  UNION

  (SELECT * FROM t1 LEFT OUTER JOIN ON t1.k = t2.k)
UPDATE Syntax

UPDATE [table_name]
SET
  [column_1] = [value_1]
  [column_2] = [value_2]
WHERE
  [condition_1]
  [(AND|OR) condition_2]
  [(AND|OR) condition_3]
  ...
  [(AND|OR) condition_n]
DELETE Syntax

DELETE FROM [table_name]
WHERE
  [condition_1]
  [(AND|OR) condition_2]
  [(AND|OR) condition_3]
  ...
  [(AND|OR) condition_n]
Query Me This

- How many tags are there of each type?
- How many authors are there of each type?
- Find the article with the most images?
- Find the article with the most tags?
- Find the most commonly applied tag?
- How many articles are authored by more than one author?
- How many articles are authored by organizations services?