GENERAL ASSEMBLY

DATA ANALYSIS: MOVING FROM EXCEL TO SQL

Nathan Halko, Data Scientist @spotright

AGENDA / LEARNING OBJECTIVES

- Install MySQL server and Workbench
- Relational database concepts and data structures
- SQL syntax and query statements
- Import / Export
- Practice questions

Source:

INSTALL MYSQL SERVER AND WORKBENCH

MYSQL SERVER

- http://dev.mysql.com/downloads/mysql/
- http://dev.mysql.com/doc/refman/5.7/en/installing.html

MYSQL WORKBENCH

- https://dev.mysql.com/downloads/workbench/
- https://dev.mysql.com/doc/workbench/en/wb-installing.html

- *Hint: Follow the installation instructions closely*

RELATIONAL DATABASE CONCEPTS AND DATA STRUCTURES

DATABASE TOOLS

- Excel
  - 1,000’s of rows
  - in-memory
- Relational (SQL) database
  - 10’s of millions of rows
  - single server
- noSQL
  - billions to unlimited
  - cluster of servers, sharded, replicated

WHERE DOES MY DATA LIVE?

Disk - persistent layer, not in memory

Server - running program to provide access

SQL - structured query language to interact with the server

Workbench - GUI to write SQL and view schema

ANATOMY OF A RELATIONAL DATABASE

- Table: collection of related data consisting of rows and columns
- Row: an individual horizontal entry in a table, record
  - primary key (unique, required)
- Field: the specific entries within a row
  - foreign key (other tables' primary keys)
- Column: vertical entity containing all values within a field
- Type: numbers and letters, ie INTEGER, DOUBLE, VARCHAR, DATE, TIME, BOOLEAN
- NULL: missing data different from 0 or “”
- Schema: map of relationships within and between tables

CRUD

Put things in:

- create CREATE
- update INSERT INTO

Take things out:

- read SELECT
- delete DELETE FROM

THINGS WE WON'T COVER

- Database design
  - schema, data types
  - normalization
- "C_UD" of crud
  - we are really just pulling things out
- Admin
  - indexing - query optimization
  - performance

SQL SYNTAX AND STATEMENTS

GOAL: SELECT … FROM … WHERE

Describes the ways we ‘get’ data from the database or ‘pulling’ out a result set.

SELECT defines what columns we want.

FROM defines the database table.

WHERE puts some restrictions on what we expect.

Get me a beer from the refrigerator that is cold.
SELECT beer FROM refrigerator WHERE temp = ‘cold’ LIMIT 1;

SELECT … FROM

SELECT returns results from a table. This is the most basic SQL command.
notice: SQL keywords are in CAPS and all statements end with ;

```
USE sakila;

SELECT * FROM actor;

SELECT first_name,last_name FROM actor;

SELECT rental_id,amount FROM payment;
SELECT first_name,username FROM staff;
SELECT DISTINCT rating FROM film;
```

SELECT … FROM … WHERE …

The results of SELECT are filtered through a WHERE clause to return only a subset of the results matching a condition.

```
SELECT * FROM actor WHERE first_name = 'johnny';

SELECT * FROM customer WHERE active = 0;

SELECT * FROM rental WHERE inventory_id = customer_id;
SELECT * FROM city WHERE city = 'El Alto';
```

WHERE [OPERATOR] …

The WHERE clause can be =, !=, <, <=, >, >=, BETWEEN, IN, LIKE, AND, OR, NOT

```
SELECT * FROM customer WHERE active != 0;
SELECT * FROM customer WHERE address_id BETWEEN 10 AND 20;
SELECT * FROM film WHERE rating IN ('R', ‘G');
SELECT * FROM city WHERE city LIKE ‘bat%';
SELECT * FROM film WHERE title LIKE ‘%devil';
SELECT * FROM film WHERE release_year LIKE '200%';

SELECT title,description,release_year FROM film WHERE rating NOT LIKE '%G%'
   AND (rental_duration < 5 OR rental_rate < 0.99);
```

PRACTICE: SELECT … FROM … WHERE

- Find all addresses that are on a ‘Street’.
- Find all districts where the city_id is less than the postal_id.
- Find the names of all G rated films you can rent for less than $3.
- Can I rent any movies about crocodiles for either $0.99 or at least 4 days?
- Find film_ids with a category_id from 3-5.
- Find all films that include a trailer.

GOAL: OPERATE ON THE RESULT SET

Once we have defined a candidate result set we might want to manipulate the data further than just returning raw records.

ORDER BY	sort results by field.
GROUP BY	collect things on rows that have a common field value.
LIMIT	return less than the full result set.
FUNCTIONS	operations on the results of queries: *min, max, avg, count, sum*

ORDER BY [ASC|DESC]

Sort result sets

```
SELECT * FROM actor ORDER BY last_name ASC;
SELECT * FROM customer WHERE active = 0
   ORDER BY length(first_name) ASC, last_name ASC;

SELECT title FROM film ORDER BY length DESC LIMIT 10;
```

GROUP BY

Group by is used in conjunction with the aggregate functions and allows us to analyze fields between rows who share a common value in another field.

SELECT groupingCol, aggFunction(anyCol) FROM table GROUP BY groupingCol

```
SELECT rating,max(replacement_cost) FROM film GROUP BY rating;

SELECT customer_id, sum(amount) FROM payment WHERE customer_id < 7
   GROUP BY customer_id;

SELECT staff_id, count(distinct customer_id) AS total_served FROM payment
   GROUP BY staff_id ORDER BY total_served DESC;
```

GOAL: JOINS !!

We’ve gone pretty far with just a single table, now we use those same techniques across multiple tables in a join.

A join takes two tables with a common column and returns ‘joined’ rows from each table where they share a common column value.

```
SELECT * FROM
   customer JOIN rental ON customer.customer_id = rental.customer_id;


think
    table = JOIN clause
then
    SELECT * FROM table
```

JOIN

What is the question to this answer?

```
SELECT DISTINCT first_name FROM
   customer JOIN rental ON customer.customer_id = rental.customer_id;
```

JOIN

What is the question to this answer?

What are the first names of people who rented movies?

```
SELECT DISTINCT first_name FROM
   customer JOIN rental ON customer.customer_id = rental.customer_id;
```

LEFT JOIN

What are the first names of people who *didn’t* rent movies?
… who didn’t rent movies from store 1?

```
SELECT count(distinct(first_name)) FROM
   customer LEFT JOIN rental ON customer.customer_id = rental.customer_id
   WHERE store_id != 1;
```

IMPORT / EXPORT

SELECT … INTO

```
SELECT first_name,last_name INTO OUTFILE ‘/tmp/actor.csv’
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    FROM sakila.actor;
```

CREATE DATABASE|TABLE

```
#DROP DATABASE n8s_db;
CREATE DATABASE n8s_db;

USE n8s_db;

CREATE TABLE my_actor (
    first_name VARCHAR(45),
    last_name VARCHAR(45),
    id INT NOT NULL AUTO_INCREMENT,
    PRIMARY KEY (id)
);
```

LOAD DATA LOCAL INFILE

```
LOAD DATA LOCAL INFILE '/tmp/actor.csv'
    INTO TABLE my_actor
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    IGNORE 1 ROWS;

SELECT * FROM my_actor;
```

INSERT INTO, DELETE FROM

```
INSERT INTO my_actor (first_name, last_name) VALUES
    ('Nathan', 'Halko'),
    ('Frankie', 'Lou');

DELETE FROM my_actor WHERE first_name = 'Nathan';
```

PRACTICE PRACTICE PRACTICE

PRACTICE QUESTIONS

- What is the longest movie that can be rented for $0.99 for 1 week?
- What is the id of the customer who has spent the most money on rentals?
- Who has rented the most movies?
- What is the full name of the actor who has been in the most films?
- Whats the first name of the last customer to rent Beauty Grease?
- Are there any non English language films?
- Which country loves Sci-Fi movies?
- What actor has generated the most rental income for store 1?
- Write your own question and share it with us!

PRACTICE QUESTIONS

- What is the longest movie that can be rented for $0.99 for 1 week?

```
SELECT * FROM film WHERE rental_rate = 0.99 AND rental_duration = 7 ORDER BY
length DESC LIMIT 1;
```

PRACTICE QUESTIONS

- What is the id of the customer who has spent the most money on rentals?

```
SELECT customer_id,sum(amount) AS total_paid FROM payment GROUP BY
customer_id ORDER BY total_paid DESC LIMIT 1;
```

PRACTICE QUESTIONS

- Who has rented the most movies?

```
SELECT customer_id, count(customer_id) AS num_rentals FROM rental GROUP BY
customer_id ORDER BY num_rentals LIMIT 1;
```

PRACTICE QUESTIONS

- What is the full name of the actor who has been in the most films?

```
SELECT first_name,last_name,count(film_actor.film_id) AS num_films FROM
   film_actor JOIN actor ON film_actor.actor_id = actor.actor_id
   GROUP BY film_actor.actor_id
   ORDER BY num_films DESC LIMIT 10;
```

PRACTICE QUESTIONS

Whats the first name of the last customer to rent Beauty Grease?

```
SELECT customer.first_name FROM
    rental JOIN inventory ON rental.inventory_id = inventory.inventory_id
    JOIN film ON inventory.film_id = film.film_id
    JOIN customer ON rental.customer_id = customer.customer_id
    WHERE title = 'Beauty Grease'
    ORDER BY rental.rental_date DESC LIMIT 1;
```

PRACTICE QUESTIONS

Are there any non Enlish language films?

```
SELECT * FROM film JOIN language ON language.language_id = film.language_id
WHERE language.name NOT LIKE 'English';
```

PRACTICE QUESTIONS

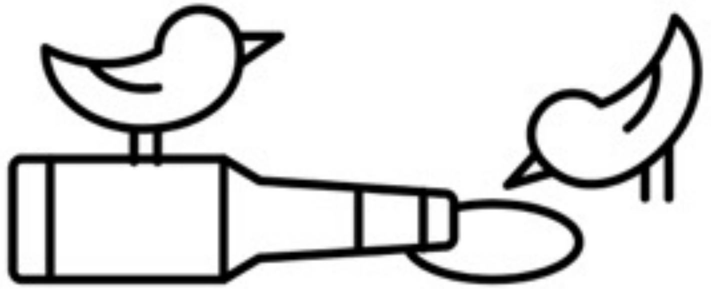
Which country loves Sci-Fi films?

```
SELECT country.country, count(rental.rental_id) AS total_rentals FROM
    category JOIN film_category ON category.category_id =
film_category.category_id
    JOIN inventory ON inventory.film_id = film_category.film_id
    JOIN rental ON rental.inventory_id = inventory.inventory_id
    JOIN customer ON customer.customer_id = rental.customer_id
    JOIN address ON customer.address_id = address.address_id
    JOIN city ON city.city_id = address.city_id
    JOIN country ON city.country_id = country.country_id
    WHERE category.name LIKE 'Sci-Fi'
    GROUP BY country.country_id
    ORDER BY  total_rentals DESC LIMIT 1;
```

Interested in Learning More?

ALE AND DATA ANALYTICS SAMPLE CLASS

- AUGUST 2ND
- 6:00–7:30



ADMISSIONS

- JULIETTE@GA.CO

GA COLOR PALETTE

🟨 **YELLOW**	255/216/0
MINT	131/237/217
TURQUOISE	30/202/199
BURGUNDY	113/10/51
PINK	255/157/182
🟥 **RED**	229/27/36
LIGHT GREY	229/229/229
DARK GREY	88/88/91
⬛ **BLACK**	0/0/0