QuantSprout

SQL Bootcamp

By Craig Sakuma

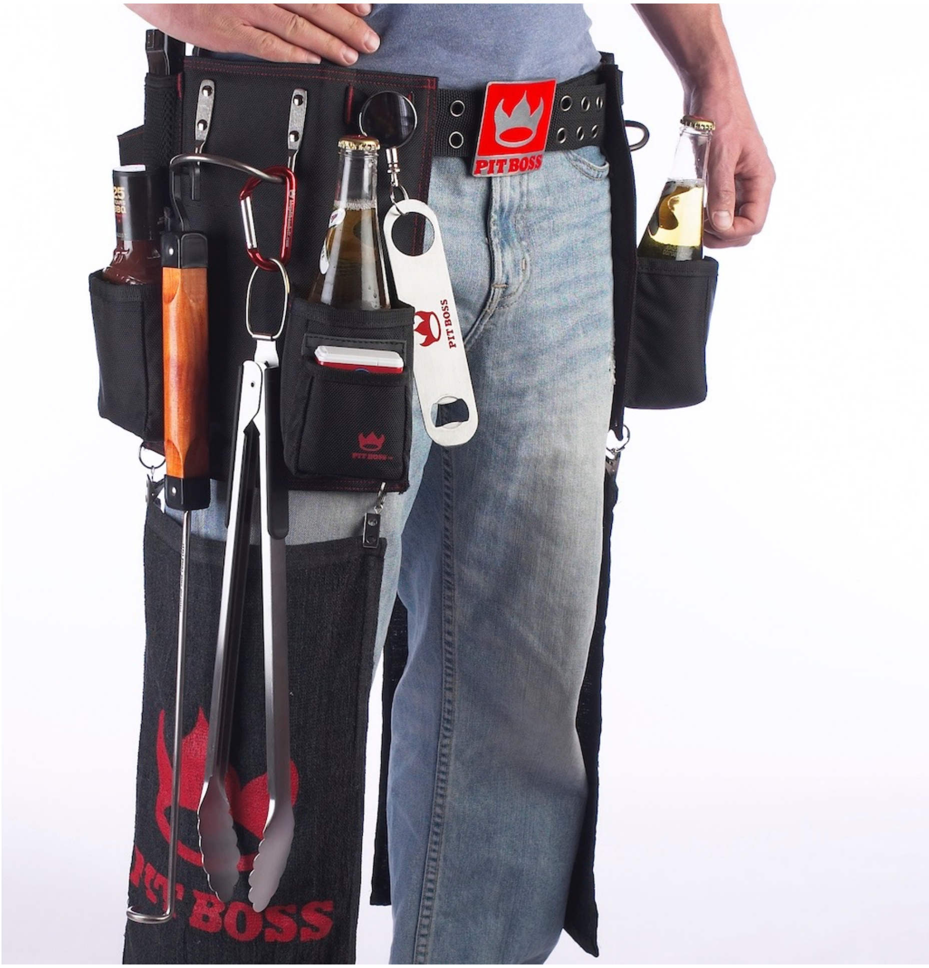
Introductions

Craig Sakuma

- Founder of QuantSprout
- General Assembly Instructor for Data Science
- MBA from Wharton
- B.Eng from Northwestern University

Fun Fact

Developed a novelty BBQ product that was featured in USA Today



Class Introductions

- Name
- What’s your job?
- How do you plan to apply skills from today’s workshop?
- Fun Fact

Objectives for Class

- Get strong foundation of SQL
- Immediately use skills at work
- Remove barriers/frustration
- Develop skills to be self-sufficient after class
  - Tools and examples for practicing on your own
  - Comfort with SQL to learn on own
  - Ability to troubleshoot problems

Course Structure

- Lectures on topics
  - Interaction is good
  - Feel free to ask questions
  - If there's not enough time to cover questions, we'll put it in a parking lot for after class

- Hands on exercises
  - Pair programming
  - Mix up partners

Schedule

Time	Topic
10:00 – 11:00	Overview of SQL
11:00 – 12:30	Fundamentals of Queries
12:30 – 1:30	Lunch
1:30 – 2:00	Advanced Queries
2:00 – 2:30	Creating Tables
2:30 – 3:45	Joining Tables
3:45 – 4:15	Functions
4:15 – 5:00	Group By

What is SQL?

- Structured Query Language
  - Programming language
  - Structured data (requires some overhead)
  - Relational Database
  - Allows you to share large sets with many users
  - Scalable data storage

SQL is Searchable

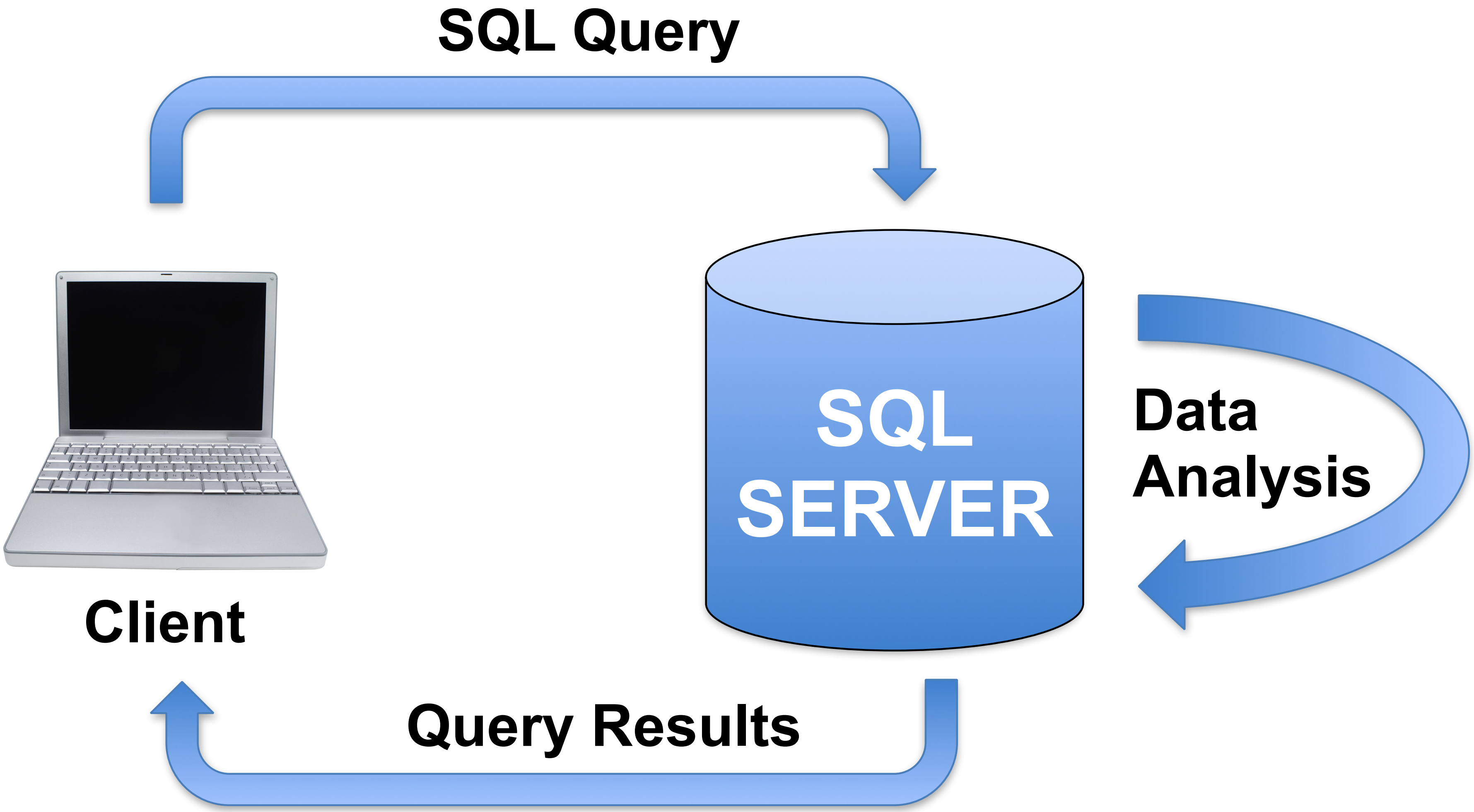
SQL Types

- PostgreSQL
- SQLite
- MySQL
- Amazon Redshift
- Oracle
- Microsoft SQL Server

Amazon Redshift: use cluster of computers vs. other SQL is storing data locally in 1 computer

Functionality is similar but syntax can be different

SQL – Behind the Scenes



SQL Query

Client

SQL SERVER

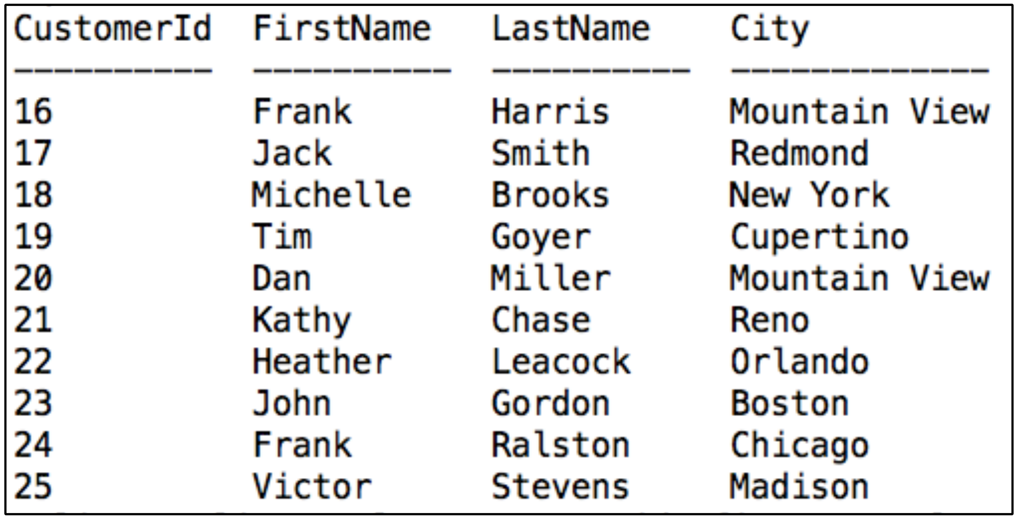
Data Analysis

Query Results

However, SQLite operates locally

Tables

- Data containers
- Organized by rows and columns



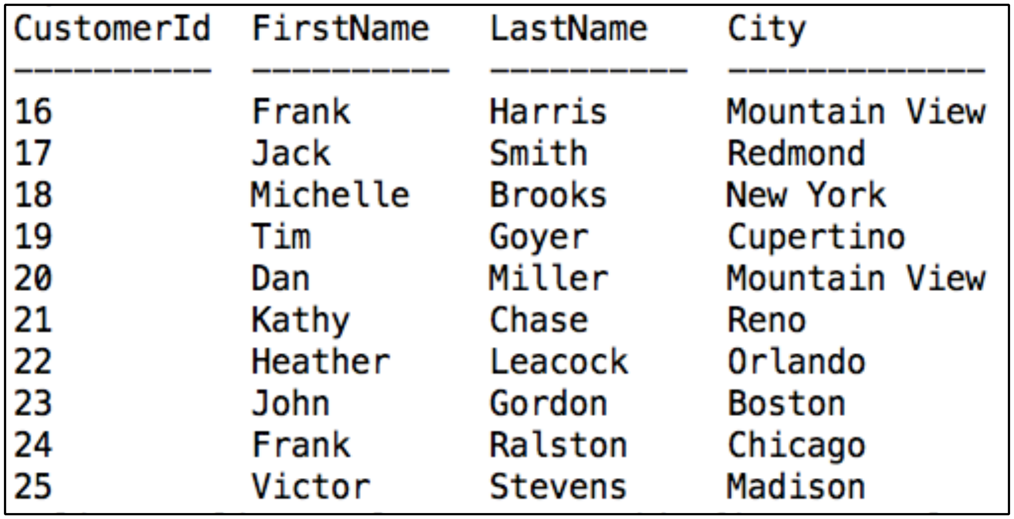
```
CustomerId  FirstName   LastName    City
----------  ----------  ----------  -------------
16          Frank       Harris      Mountain View
17          Jack        Smith       Redmond
18          Michelle    Brooks      New York
19          Tim         Goyer       Cupertino
20          Dan         Miller      Mountain View
21          Kathy       Chase       Reno
22          Heather     Leacock     Orlando
23          John        Gordon      Boston
24          Frank       Ralston     Chicago
25          Victor      Stevens     Madison
```

Tables Are Like Spreadsheets

Tables

Data Fields
(like columns)

Database Records
(like rows)



```
CustomerId  FirstName   LastName    City
----------  ----------  ----------  -------------
16          Frank       Harris      Mountain View
17          Jack        Smith       Redmond
18          Michelle    Brooks      New York
19          Tim         Goyer       Cupertino
20          Dan         Miller      Mountain View
21          Kathy       Chase       Reno
22          Heather     Leacock     Orlando
23          John        Gordon      Boston
24          Frank       Ralston     Chicago
25          Victor      Stevens     Madison
```

- Fields have Data Types
- Primary Keys (unique identifier)
- Foreign Keys (identifier for linking to other tables)
- Fields frequently are both Primary and Foreign Keys

Data Types

- Text
  - char(***n***)
  - varchar(***n***)
  - text
- Numbers
  - integers
  - floats
- Boolean (True/False)
- Temporal
  - dates
  - times
  - timestamps

varchar(n) is efficient with memory e.g. when 20/100 char spaces are used, remaining 80 memory is re-allocated

BUT

if character data is all of equal length, then char(n) is ok

Sample Database – sql_bootcamp.db

11 Data Tables

- Album
- Artist
- Customer
- Employee
- Genre
- Invoice
- InvoiceLine
- MediaType
- Playlist
- PlaylistTrack
- Track

QuantSprout

Note: Database example come from ChinookDatabase 1.4

Syntax for Slides

Brackets are placeholders

Example (for change directory):

```
cd <directory name>
```

Applied for changing to directory ‘SQL’:

```
cd SQL
```

SQL Code vs. SQLite Commands

SQLite Commands

- Start with a period like .quit
- Used for opening databases, reading files, changing settings
- Case sensitive
- Doesn’t require a semi-colon at end of statement

SQL Code

- Universal code used for all versions of SQL
- Not case sensitive
- Code requires a semi-colon at end of statement because SQL commands can be written across multiple lines

Launch SQLite3

All Users:

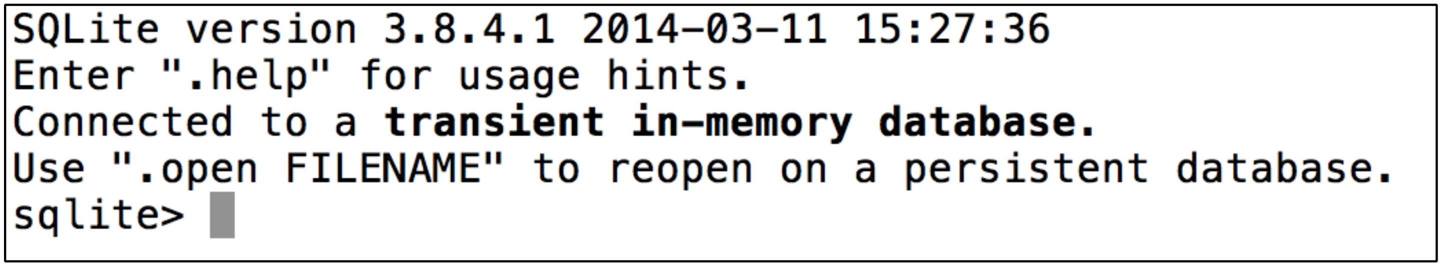
- Create a folder on your desktop called SQL
- Move the sql_bootcamp.db file into the new folder

Mac Users:

- Launch your terminal
- Change directories to your SQL folder by entering “cd Desktop/SQL”
- Enter “sqlite3”

Windows Users:

- Move the sqlite3 executable file into same folder as sql_bootcamp.db file
- Double click your sqlite3 executable file



```
SQLite version 3.8.4.1 2014-03-11 15:27:36
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> 
```

SQLite3 – Open a Database

Open a database:

```
.open sql_bootcamp.db
```

View the database schema:

```
.schema
```

Explore the Data Tables and Identify Foreign Keys

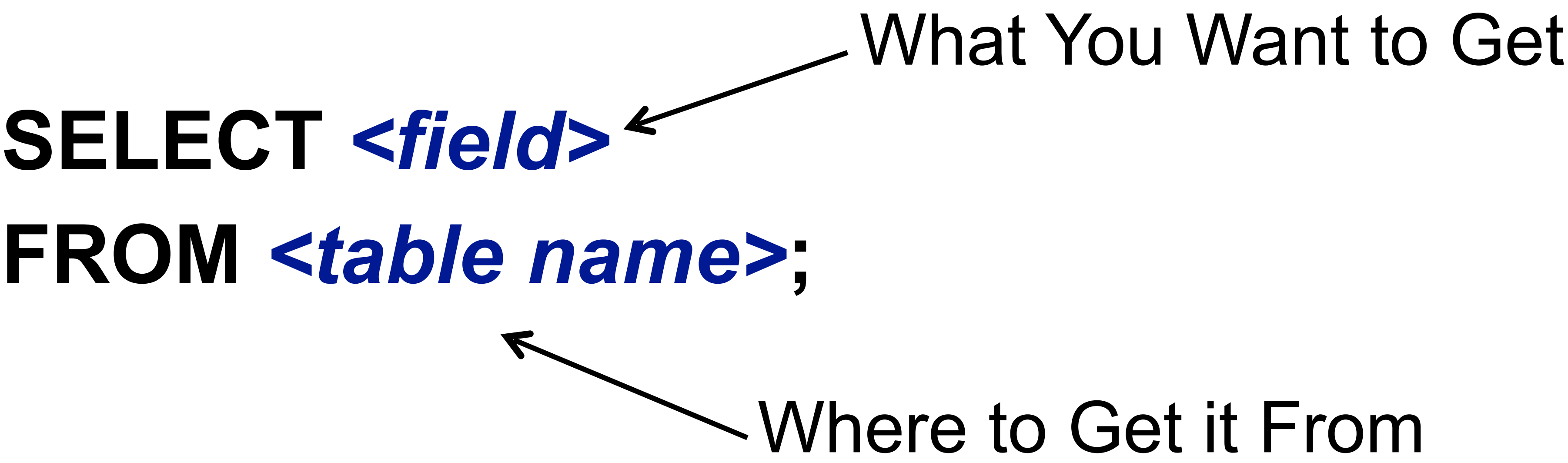
Query Types

- SELECT
  - Creates view of records from database
- INSERT
  - Inserts new records into a table
- UPDATE
  - Updates existing records into a table
- DELETE
  - Removes records

CHANGES DATA IN TABLES

Simplest Query

You Only Need Two Things for a Query:



What You Want to Get

```
SELECT <field>
FROM <table name>;
```

Where to Get it From

Semi-colon is used to mark the end of a query

Basic Query Structure

- SELECT (values you want to view)
- FROM (data source – table name)
- WHERE (conditions for results)
- ; (delimiter for end of query)

Common Basic Queries

```
SELECT <field 1>, <field 2>
FROM <table name>
WHERE <field 1> = ‘value’;
```

Select multiple fields

Create conditions for query

You Can Create Multiple Conditions with AND /OR

Queries with Limits

```
SELECT *
FROM <table name>
WHERE <field 1> = 5
LIMIT 10;
```

Wildcard for Selecting All Fields

single quote needed only for dates and character values

Limit Results to 10 Records

Avoid Requesting Too Much Data
And Getting a SQL “Time Out”

Why Use a Text Editor

- Save your queries for future reference
- Easier to edit when you make mistakes (especially when you have large queries)
- Include multiple queries in single file
- Comment out queries you don’t want to execute

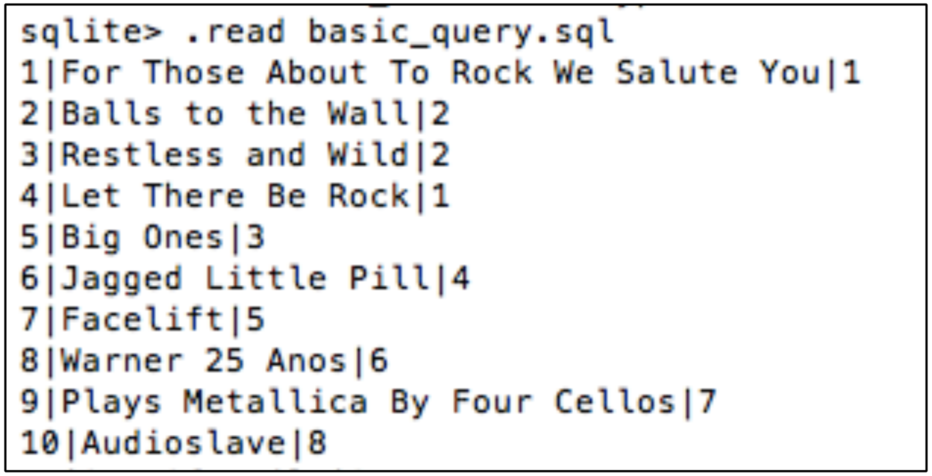
SQLite3 – Read SQL File

Execute SQL from file

.read *<filename>*

Try it yourself:

.read class_exercises.sql



```
sqlite> .read basic_query.sql
1|For Those About To Rock We Salute You|1
2|Balls to the Wall|2
3|Restless and Wild|2
4|Let There Be Rock|1
5|Big Ones|3
6|Jagged Little Pill|4
7|Facelift|5
8|Warner 25 Anos|6
9|Plays Metallica By Four Cellos|7
10|Audioslave|8
```

SQLite3 Settings

View Column Headers

.header on

Organize Data in Columns:

.mode column

(Note: default mode is list)

Makes Viewing Results
Easier to Read

SQLite3 – Comment

Write comments that aren't executed as code

-- *<single line of text>*

/* *<multiple lines of text>*

***<multiple lines of text>* */**

Use for documenting code

Comment out code that you don't want to run

sublime text shortcut: command + /

Instructions for Exercises

- Pair programming
  - Using only one computer
  - Take turns typing
  - Collaborate on solutions
- Resources
  - Stackoverflow / Google
  - Don’t be afraid to ask for help

Exercise #1

1. What are the genres in the database?

2. What are the customer names that are from California?
   (Hint: text strings need to be in single quotes)

Bonus: Explore data samples from the rest of the tables in the database

SELECT Options

- SELECT COUNT(*)
  Counts the records in the database

only counts non-null values

- SELECT DISTINCT firstname
  Selects unique first names
  (Note: if multiple fields are provided it will select distinct combinations of those fields)

- SELECT COUNT(DISTINCT firstname)
  Counts the number of unique first names

WHERE Clauses

- Combine multiple conditions with AND / OR

```
SELECT *
FROM customer
WHERE state=‘CA’ OR state=‘WA’ OR state=‘OR’;

WHERE country!=‘USA’ AND country!=‘Canada’;
```

Apply Any Equality condition: >, >=, <, <=, =, !=

WHERE Clauses

- IS NULL / IS NOT NULL
  WHERE genreid IS NULL

IS can handle null values but = can not handle null values

- IN (‘item1’,’item2’, etc….)
  WHERE state IN (‘NY’, ’PA’, ‘MD’, ‘DE’)

- BETWEEN ***<start>*** AND ***<end>***
  WHERE milliseconds BETWEEN 180000 AND 240000

range is inclusive of endpoints and can be used with numerical values as well as date type values

Best Practices

- CAPITALIZE SQL COMMANDS
- Use Indentation to Improve Readability:

Option #1

```
SELECT
      albumid,
      title
FROM
      album
```

Option #2

```
SELECT albumid, title
FROM album
WHERE albumid =1
```

- Error Tracking
  - Create a text file to keep notes on your errors
- Save Class Examples as Files

Exercise #2

1. How many songs are longer than 10 minutes?
2. How many invoices were there between January 1, 2010 and February 1, 2010 (hint: dates are in single quotes and use google to find format)?
3. How many tracks have a NULL composer?
4. How many distinct album titles are there? How many distinct album IDs? Why would these have different counts?

ORDER BY

- Sorts data
  - ASC for ascending
  - DESC for descending
- Can have multiple fields
  - First field listed takes precedence
- Example:

```
SELECT *
FROM track
ORDER BY genreid ASC, milliseconds DESC
```

LIKE / NOT LIKE

Value		Pattern	Result	
'abc'	LIKE	'abc'	TRUE	
'abc'	LIKE	'c'	FALSE	
'abc'	LIKE	'a%'	TRUE	% symbol is a any-length wild card
'abc'	LIKE	'_b_'	TRUE	underscore is a single character wild card

Example:

WHERE email LIKE ‘%@%._ _ _’

LIKE is not case sensitive in sqlite3

Exercise #3

1. What are the 5 longest tracks?
2. R.E.M. has collaborated with a couple artists, can you find which artists they've collaborated with?
   (Hint: Use the Artist Table)
3. How many love songs are there?
   (Hint: Use the Track Name)

Table Commands

- CREATE TABLE
  - Creates new table
  - Requires fields to be defined by data type
- DROP TABLE
  - Deletes table from database
- ALTER TABLE
  - Change database column name
  - Add or remove columns
  - Change table name

Create Table

Creating new tables requires definitions for fields and data types:

```
CREATE TABLE <table name> (
    <field name1>   <data type1>,
    <field name2>   <data type2>,
    etc…
);
```

Create Table Example

```
CREATE TABLE favorite_songs (
    id INTEGER,
    title VARCHAR(128),
    seconds INTEGER
);
```

Commit Changes

Many database require confirmation after changes are made

- Create, Alter or Drop Tables
- Insert, Update or Delete Values

Enter “COMMIT;” after changes are made

However, SQLite automatically commits

Insert Query

Use Insert Queries to add records to your tables

```
INSERT INTO <table name>
(<field 1>, <field 2>)
VALUES (<value 1>, <value 2>);
```

Insert Query Example

```
INSERT INTO favorite_songs
(id, title, seconds)
VALUES (0, ‘Call Me Maybe’, 193);
```

Drop Table

Use DROP TABLE to completely remove a table:

DROP TABLE ***<table name>***;

DROP TABLE favorite_songs;

What are JOINS?

- Technique for merging data together from multiple tables
- Tables must have a shared Foreign Key to be joined together
- Similar to vlookup formula in Excel
- Multiple JOINS can be created to merge data from more than one table
- Data from JOINS only exist in memory during the transaction of the query

How JOINS Work

- Data is merged between two tables at a time
- Cartesian product is created between two tables (fancy for every possible combination of records)
- Results are filtered based on a condition where Foreign Keys equal each other
- All fields from both tables are included in JOINS

JOIN Simulation

Location Table

Name	State
Tim	CA
Bob	NY
Jen	AZ

Age Table

Name	Years
Tim	30
Jen	20

JOIN ON location.name = age.name

Create Every Combination

Location.Name	Location.State	Age.Name	Age.Years
Tim	CA	Tim	30
Tim	CA	Jen	20
Bob	NY	Tim	30
Bob	NY	Jen	20
Jen	AZ	Tim	30
Jen	AZ	Jen	20

Filter for location.name = age.name

Location.Name	Location.State	Age.Name	Age.Years
Tim	CA	Tim	30
~~Tim~~	~~CA~~	~~Jen~~	~~20~~
~~Bob~~	~~NY~~	~~Tim~~	~~30~~
~~Bob~~	~~NY~~	~~Jen~~	~~20~~
~~Jen~~	~~AZ~~	~~Tim~~	~~30~~
Jen	AZ	Jen	20

Results of Query

Location.Name	Location.State	Age.Name	Age.Years
Tim	CA	Tim	30
Jen	AZ	Jen	20

SQL Code for Example:

```
SELECT *
FROM location
JOIN age
ON location.name = age.name;
```

This is an inner join and only outputs the matches

JOIN Syntax

```
SELECT <field>
FROM <table 1>
JOIN <table 2>
ON <table1 foreign key> = <table2 foreign key>
```

JOIN Example

```
SELECT *
FROM track
JOIN mediatype
    ON track.mediatypeid =
         mediatype.mediatypeid
LIMIT 10;
```

Need to use table name with field name since the same field can be in multiple tables

Multiple Joins

- Joins are executed one by one in the order they are written
- Output of first join is used in subsequent join
- Each join adds extra work for the server to execute (slows down query performance)
- Foreign keys must be shared between the table being joined and the previous tables that were joined

Multiple Joins - Example

```
SELECT COUNT(*)
FROM track
JOIN album
    ON track.albumid = album.albumid
JOIN artist
    ON album.artistid = artist.artistid
WHERE artist.name LIKE ‘a%’;
```

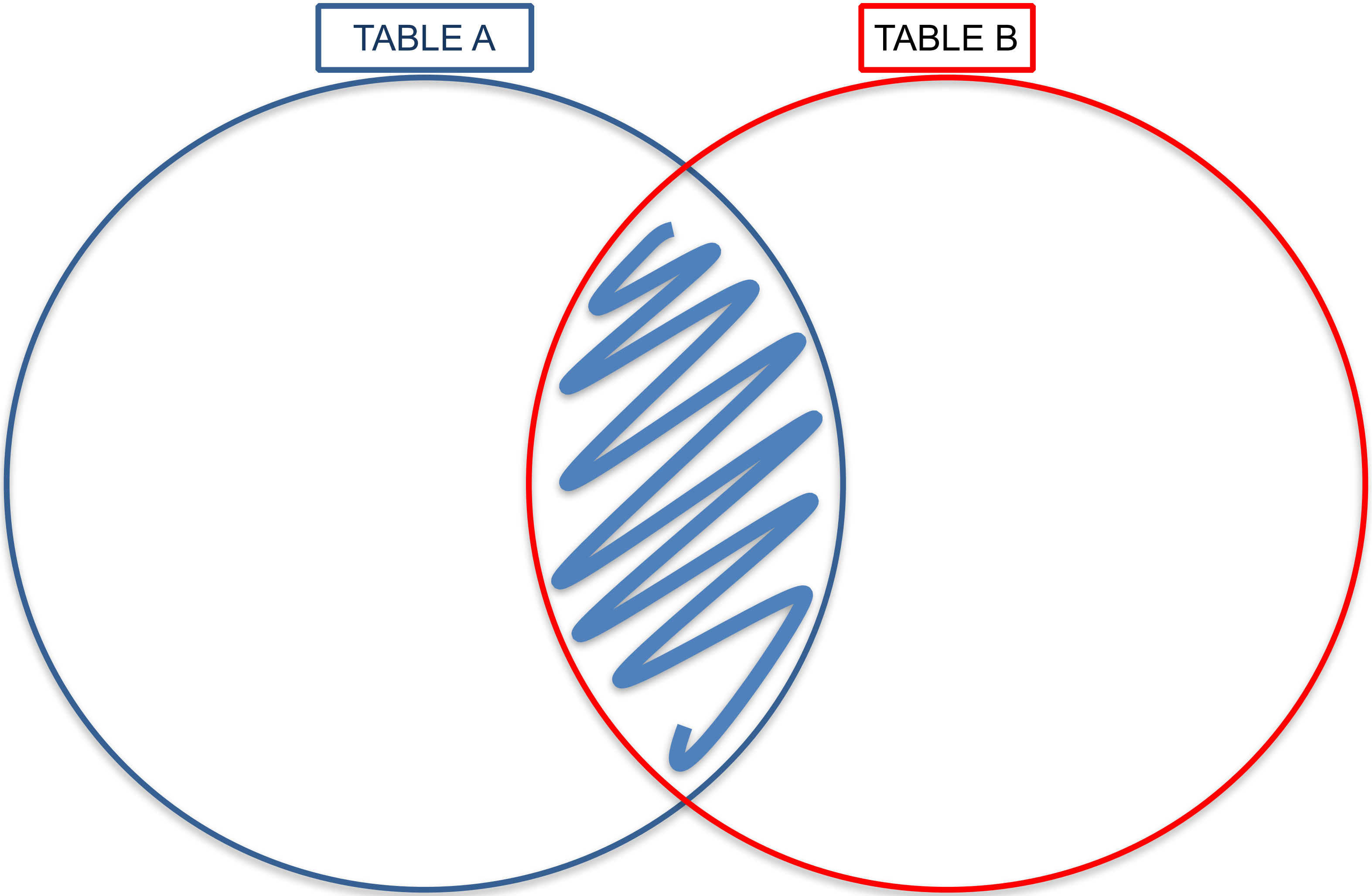
Order of Query Clauses

```
SELECT *
FROM track
JOIN genre
    ON track.genreid = genre.genreid
WHERE genre.name = ‘Rock’
ORDER BY track.name
LIMIT 10;
```

Exercise #4

1. How many tracks are rock or alternative?
2. What are the 5 longest songs (hint: use mediatype table)?
3. How many tracks are performed by R.E.M.?
4. What other interesting queries can you create that join 2 tables?

INNER JOIN



This is the default type for ‘JOIN’

QuantSprout

INNER JOIN Example

Location Table

Name	State
Tim	CA
Bob	NY

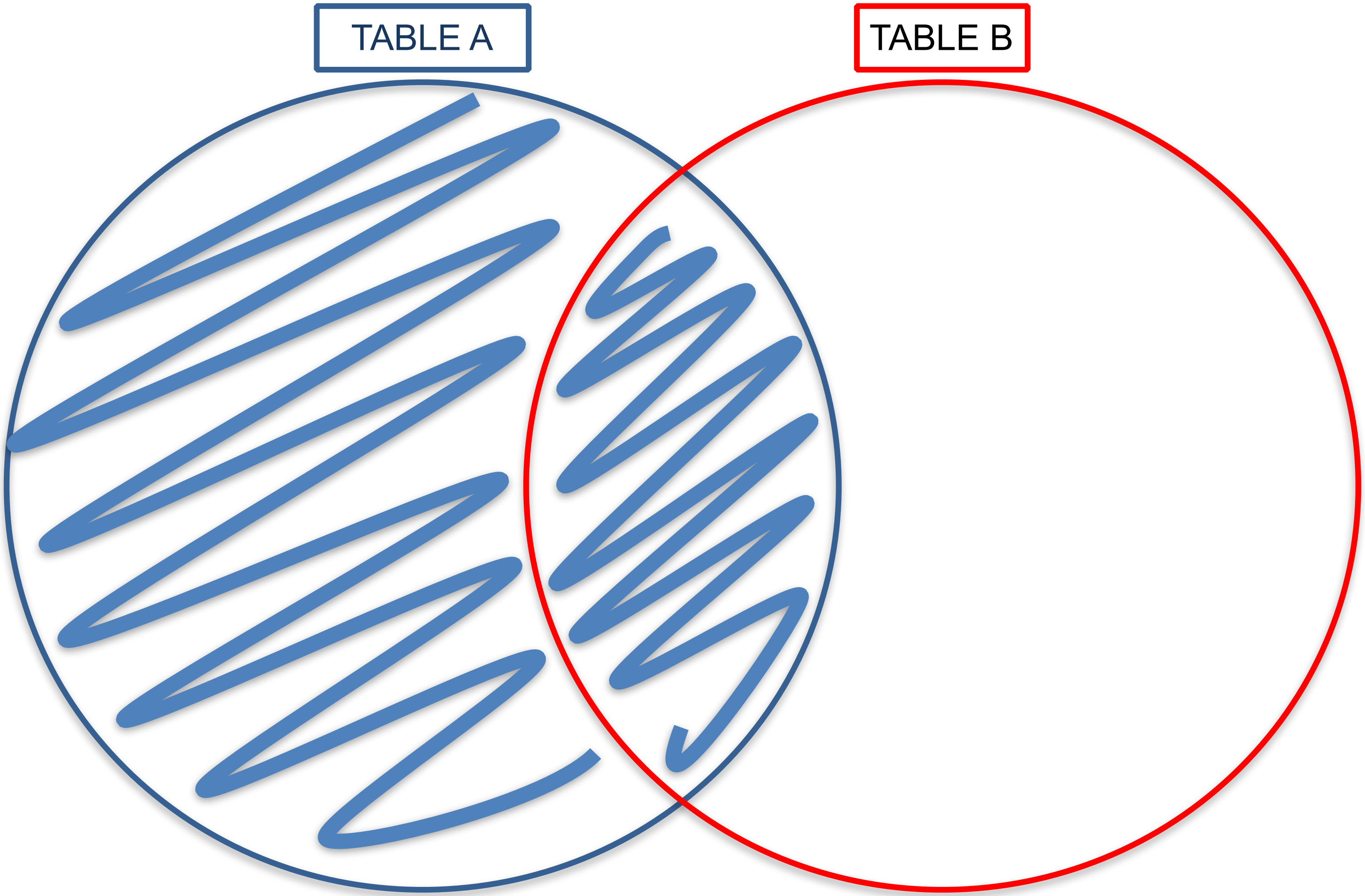
Age Table

Name	Years
Tim	30
Jen	20

Result

Location.Name	Location.State	Age.Name	Age.Years
Tim	CA	Tim	30

LEFT JOIN



QuantSprout

LEFT JOIN Example

Location Table

Name	State
Tim	CA
Bob	NY

Age Table

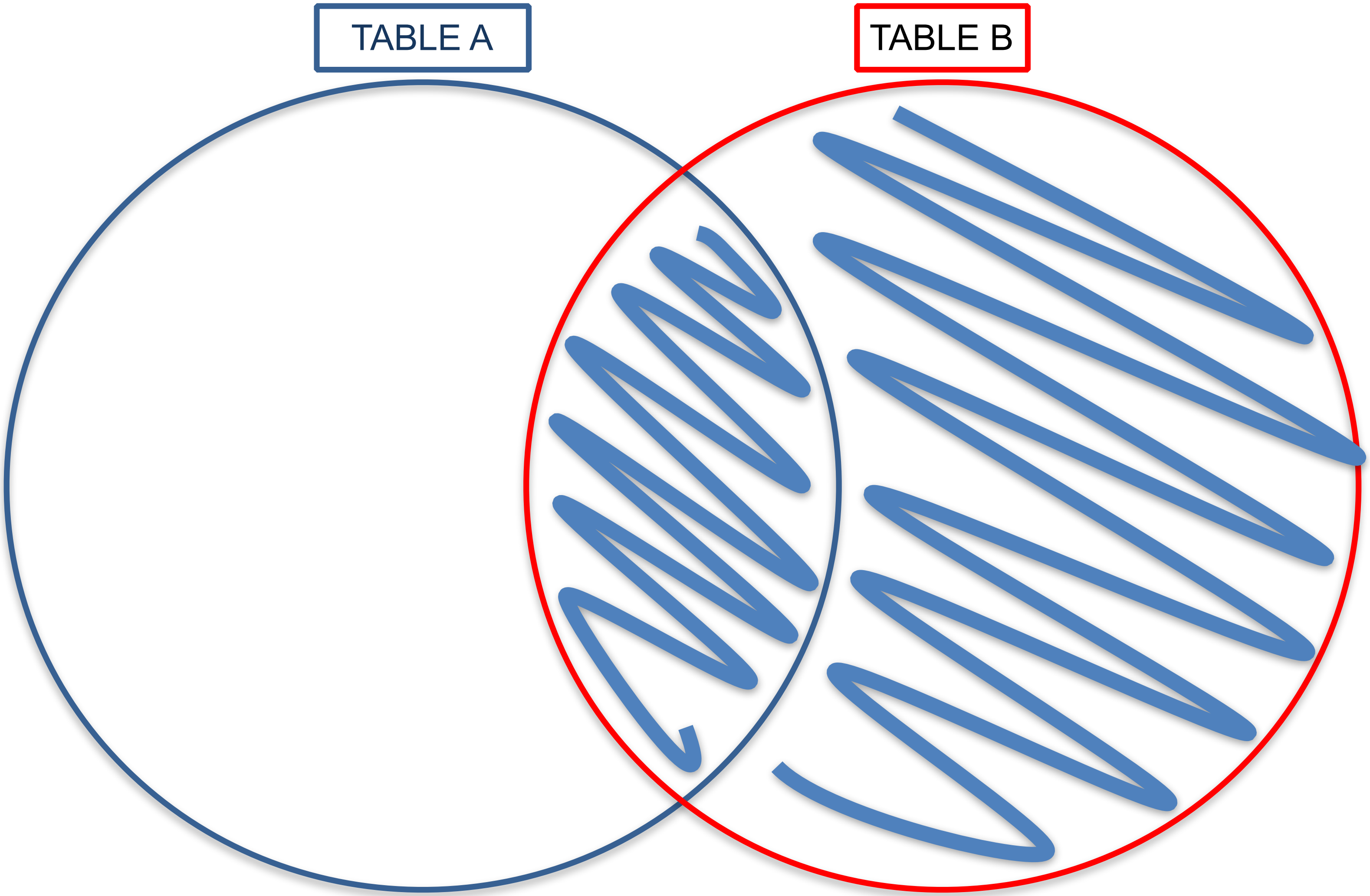
Name	Years
Tim	30
Jen	20

Result

Location.Name	Location.State	Age.Name	Age.Years
Tim	CA	Tim	30
Bob	NY		

Common case for Left joins is a star schema (where one central table links each of the other tables but none of the other tables relate to each other). Here doing a series of left joins ensures that you do not drop anything from the main table.

RIGHT JOIN



QuantSprout

RIGHT JOIN Example

Location Table

Name	State
Tim	CA
Bob	NY

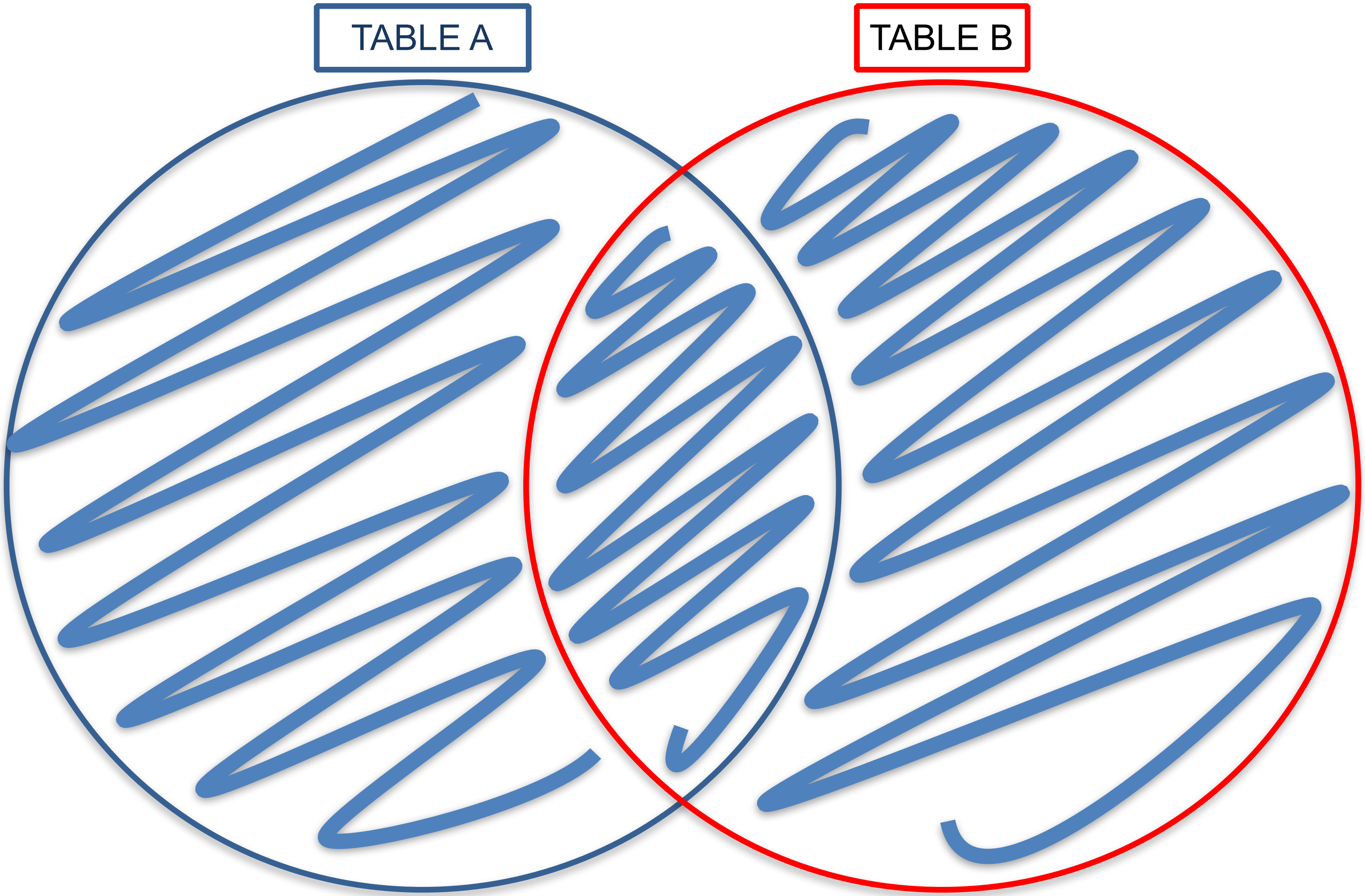
Age Table

Name	Years
Tim	30
Jen	20

Result

Location.Name	Location.State	Age.Name	Age.Years
Tim	CA	Tim	30
		Jen	20

OUTER / FULL JOIN



QuantSprout

FULL JOIN Example

Location Table

Name	State
Tim	CA
Bob	NY

Age Table

Name	Years
Tim	30
Jen	20

Result

Location.Name	Location.State	Age.Name	Age.Years
Tim	CA	Tim	30
Bob	NY		
		Jen	20

Aggregation Functions

- SELECT SUM(amount)
- SELECT MAX(release_year)
- SELECT MIN(length)
- SELECT AVG(price)

You can select multiple functions in same SELECT clause

Aliases

- Fields can be renamed:

  ```
  SELECT milliseconds/1000.0 AS seconds
  ```

- Tables can be given aliases:

  ```
  SELECT t.name, t.milliseconds, g.name
  FROM track t
  JOIN genre g
  ON t.genreid = g.genreid;
  ```

  Alias must be used consistently because once the alias is built sql will no longer recognize the original name

Exercise #5

1. What was the sales total for January 2010?

2. What is the average length of a song by R.E.M.? (Convert results to minutes, use aliases for tables, and rename results as minutes)

Group By

- Technique for Aggregating Data
- Usually requires aggregation function in SELECT statement
- Similar to Pivot Tables

Group Example

Name	City
Tim	San Francisco
Jen	Los Angeles
Bob	New York
Katie	San Francisco
Dylan	San Francisco
Sia	New York

What would you do if you wanted to Group by City?

Group Example

Name	City
Tim	San Francisco
Katie	
Dylan	
Jen	Los Angeles
Bob	New York
Sia	

Rows are sorted by city and separated into groups

Group Example

Name	City
Tim	San Francisco
Katie	
Dylan	
Jen	Los Angeles
Bob	New York
Sia	

How would you describe each group with a single value?

Group Example

Name	City	Count
Tim	San Francisco	3
Katie		
Dylan		
Jen	Los Angeles	1
Bob	New York	2
Sia		

Aggregation functions
(Count, Sum, Average, Max, Min)

GROUP BY Syntax

```
SELECT <group by field>,
       <aggregation function> (<field>)
FROM <table name>
GROUP BY <group by field>;
```

If this is not included, you won't know which group corresponds to your grouped aggregate

GROUP BY Example

```
SELECT composer, COUNT(*)
FROM track
GROUP BY composer;
```

HAVING

- Feature for only GROUP BY
- Similar to WHERE clause

where clause filters before the aggregation function

- Provides ability to apply conditions to the aggregation functions
- Example:

```
SELECT composer, COUNT(*)
FROM track
GROUP BY composer
HAVING COUNT(*) > 20;
```

Having filters after the aggregation happens

QuantSprout

GROUP BY Example

- For example:

```
SELECT composer, COUNT(*)
FROM track t
JOIN genre g
    ON t.genreid = g.genreid
WHERE g.name LIKE ‘%alternative%’
GROUP BY composer
HAVING COUNT(*) >20
ORDER BY COUNT(*) DESC
LIMIT 10;
```

Exercise #6

1. Which Artists have the most Tracks?
2. Which Albums have the longest playing time?

Bonus: How does the answer for #2 change if you limited the results to music?

ADDITIONAL TECHNIQUES

SUBQUERIES

- Use results of one query as an input to another query
- Powerful, but can also add complexity
  - Less intuitive to read
  - Harder to trouble shoot when errors occur
- Build and test the subquery first
- Use WITH AS to create an alias for the subquery

EXTRACT from Date

- Components of Dates can be Extracted
  - day, month, year
  - hour, minute, second
- SELECT EXTRACT(month from InvoiceDate) FROM Invoice;

CASE STATEMENTS

- Similar to IF statements in Excel
- Create new values from existing data
- For example:
  - You have customer age data
  - Customers behave in age segments (e.g., kids, teens, adults, seniors)
  - CASE statements can be used to create categories for age ranges

Resources for Future Reference

- W3 Schools SQL Tutorials
  www.w3schools.com/sql

- Tutorials Point
  tutorialspoint.com/sql

QuantSprout

yelp.com/biz/quantsprout-san-francisco