STAT 471: Introduction to R Programming Lecture

Lecture 8: Introduction to SQL, Relational Databases, and Applications in R

Cory Suzuki

Department of Mathematics & Statistics
California State University, Long Beach

6 October 2025

Overview

1. Introduction to Databases and SQL
2. Introduction to Structured Query Language (SQL)
3. Applications of SQL in R

There are overlaps that we'll see from the dplyr package in SQL!

What are Relational Databases?

dplyr is helpful in R, but SQL is better for cloud computing technologies & databases.

Recall that in R, we used dplyr to manipulate and transform datasets. Let's use relational databases and SQL within R to create and transform data. Relational databases relate data between tables and store them for data manipulation and transformation.

- Tables represent entities, such as customers, departments, etc. usually referring to nouns or objects.
- Columns and rows are represented by attributes within a table
- Tables can be connected via relationships. We will be taking a look at a few relationships and represent them in a visual way with Unified Modeling Language (UML) and cardinality.

depts., dept. chairs

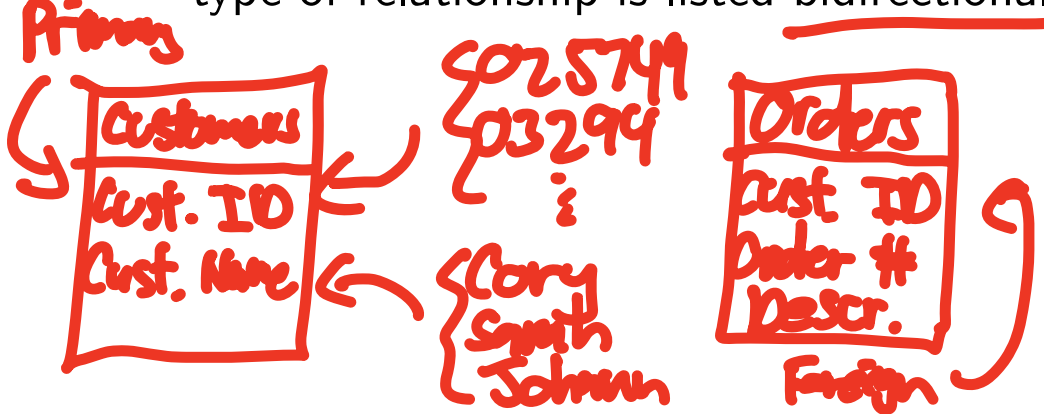
Table { Columns → (attributes) / Rows } → Relational DB

Unified Modeling Language (UML)

Visualizing relationships between tables

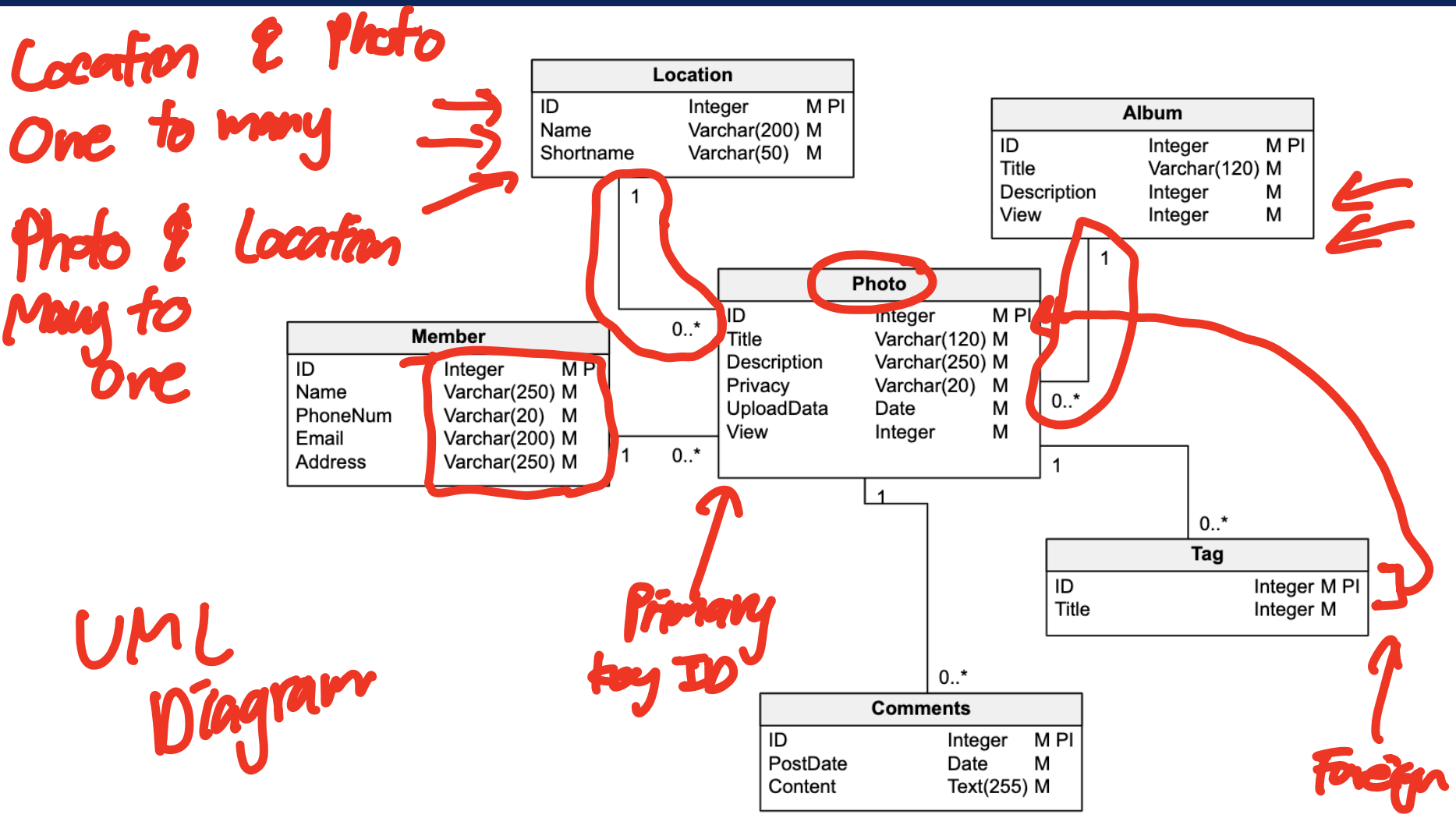
- UML diagrams model relationships between structures such as databases and object-oriented coding practices. (Java) → Classes (OOP)
- Tables are usually represented in vertical rectangles and have attributes and the respective datatypes listed within the rectangles.
- Tables are connected by lines, and at each end of the line, the cardinalities or the type of relationship is listed bidirectionally

$|\{1,2,3\}| = 3$



In SQL and UML, cardinality refers to table relationships, not size

Relational Database UML Example



Unified Modeling Language (UML)

- UML diagrams model relationships between structures such as databases and object-oriented coding practices.
- Tables are usually represented in vertical rectangles and have attributes and the respective datatypes listed within the rectangles.
- Tables are connected by lines, and at each end of the line, the cardinalities or the type of relationship is listed bidirectionally
- 1..1 cardinality represents a one-to-one relationship (example: Person and SSN)
- 1..* cardinality represents a one to many relationship (example: Book Club and Club Members) 1.. 0*
- *..* represents a many to many relationship (example: students taking classes at CSULB
- Primary keys uniquely identify each row in a table, foreign keys refer to the primary keys in another table (Customer ID for example)

SSN is another example of a primary key

Attribute Datatypes

- INTEGER are integer values
- DECIMAL(p, s) are decimals/floats where p is your precision and s is your scale
- FLOAT are floating point values with no precision or scale specified
- VARCHAR(n) are strings/characters where n represents the maximum length
- DATE, TIME, YEAR are dates, times and year types respectively
- BOOLEAN are either true or false (think about logicals in R
- Example: phone numbers can be represented as VARCHAR(20)
- Example: Sales may be represented as FLOAT or DECIMAL(2)

Connecting the Dots, or More Like Tables

Now with these concepts laid out to help us create and modify database structures, we now need a way to translate this to a language that the computer can understand. This is where Structured Query Language (SQL) comes in!

Queries written by us

→ Allow us to create tables, attributes, and modify existing rows of data

A Tale of Two Paradigms: Data Manipulation Language (DML) and Data Definition Language (DDL)

Creating table or column → DDL Modifying data, adding rows → DML

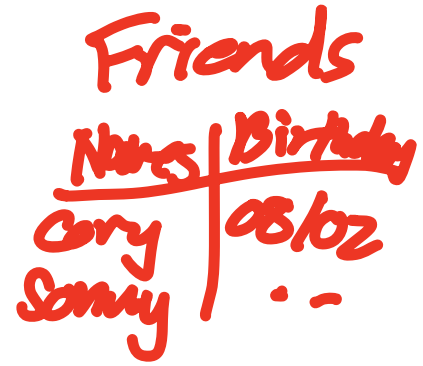
- There are two areas of applying SQL, we can create data and databases, but we can also modify existing data and databases as well
- Data Definition Language (DDL) are SQL keywords that create data and database structures/tables, such as: CREATE (creating a table), ALTER (adding, deleting, or changing columns), TRUNCATE (removing all rows but not deleting the table itself), DROP (Deletes table along with the structure), and RENAME (renaming a table)
- Data Manipulation Language (DML) are SQL keywords that modify the data and structures within databases.

Data modification or retrieval

DROP customers;

Data Manipulation Language Basics

- Notice that some of these have their respective dplyr counterparts!
- SELECT selects attributes/columns from a table
- INSERT adds rows into attributes within a table
- DELETE removes rows within a table
- UPDATE modifies rows within attributes in a table
- Example: SELECT name, birthday FROM friends
- Example: INSERT INTO friends (name, birthday) VALUES ('Charlton Suzuki', '12-04-1961')
- Note that SQL keywords are traditionally capitalized to make SQL code readable, but it isn't required



Data Query Language Keywords

- DISTINCT returns distinct rows and removes duplicates
- COUNT returns the number of rows that meets a specific condition
- SUM calculates the total sum of a numeric column
- MAX returns the largest numeric value in a column
- MIN returns the smallest numeric value in a column
- AVG calculates the mean of a numeric column
- ORDER BY returns rows in a particular order, ASC for ascending order and DESC for descending order
- WHERE is used to specify conditions
- GROUP BY groups your rows by categories
- HAVING is similar to WHERE except that it is used in conjunction with GROUP BY

In R, mean()

MEDIAN calculates med

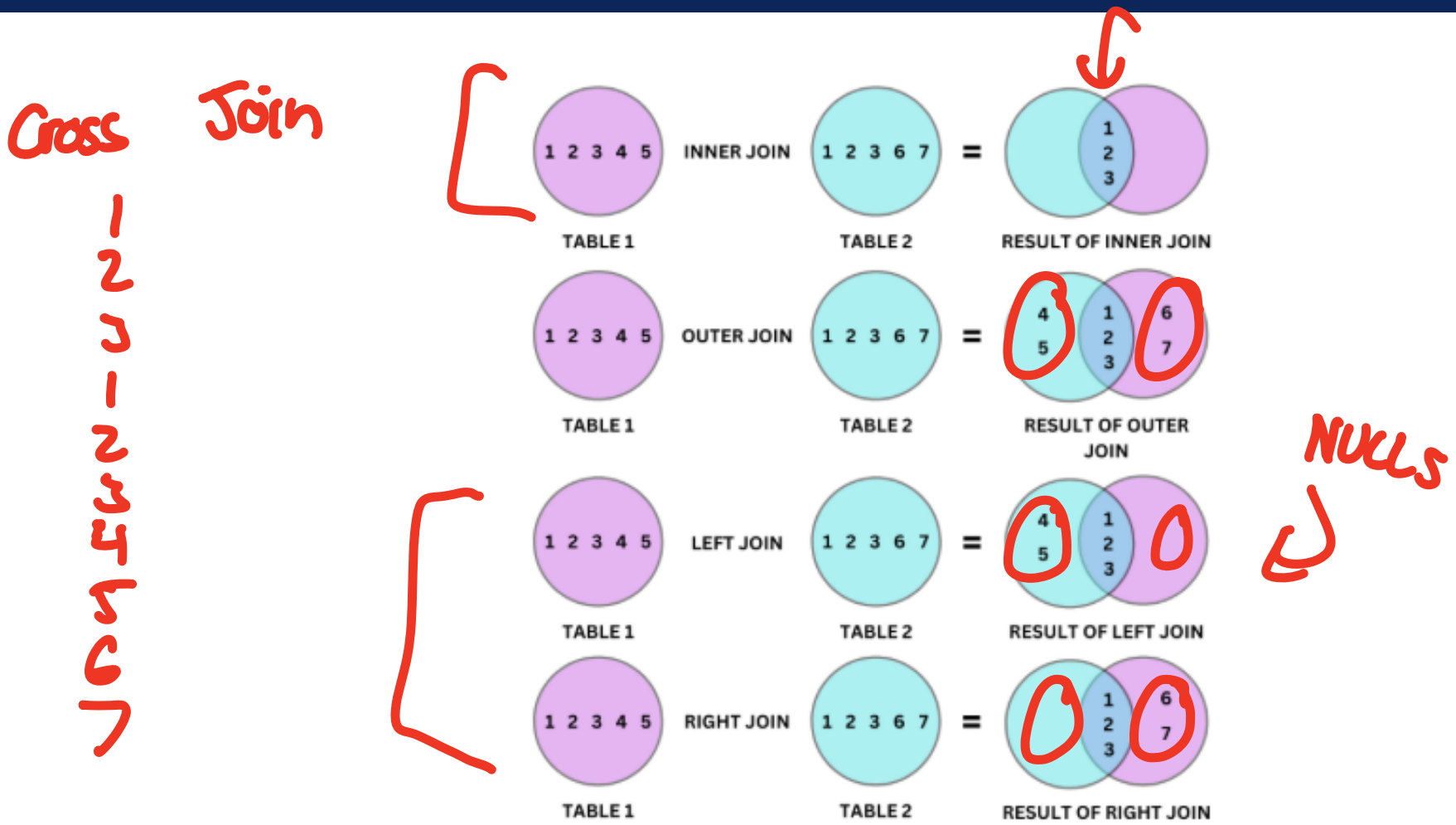
conditional WHERE rooms >= 5

GROUP BY first, and then use HAVING

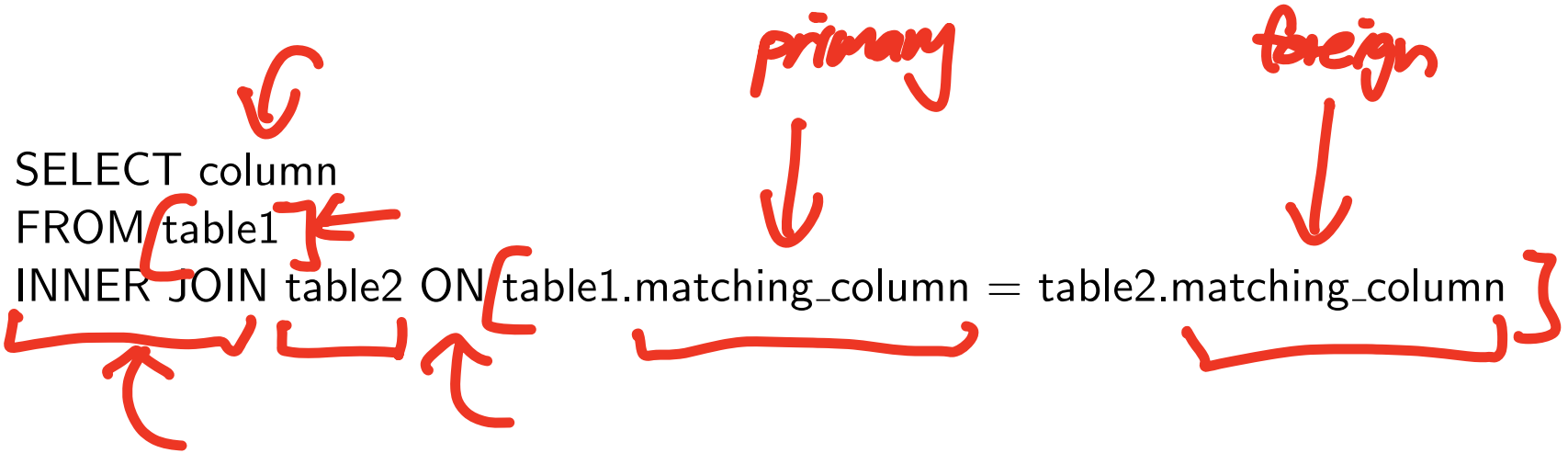
Types of Joins

- To join tables in a relational database, we can use special keywords called joins
- To better picture joins, you might want to recall elementary set theory principles via Venn Diagrams
- INNER JOIN joins tables based on the presence of primary and foreign keys. Look for what attributes tables have in common, and that's your key. Rows with matching values will be returned. $A \cap B$
- LEFT JOIN returns all rows from the left table, along with matching rows from the right table. If there is no match, NULL values are returned for columns from the right table.
- RIGHT JOIN is very similar to LEFT JOIN for the rows for which there is no matching row on the left side, the result-set will contain null.
- Unlike an inner join, an outer join includes non-matching rows from one or both tables, filling in NULL values for the columns of the table where no match exist.

Visualization of Joins



SQL Joins Syntax



Note that this syntax can be used for different types of joins except cross joins (these joins act as Cartesian products and combine all data from both tables, but we will not cover cross joins as they aren't too useful in data science)

table1 AND table2

FROM table2
JOIN table1 ON ...

SQL Query Examples Table

Note that when we write queries in SQL, we need ";" at the end of every query. Let's use this table to do some examples:

airbnb_listings

id	city	country	number_of_rooms	year_listed
1	Paris	France	5	2018
2	Tokyo	Japan	2	2017
3	New York	USA	2	2022

SQL Query Examples 1

(a) Write an SQL query that retrieves all columns from the airbnb_listings table.
(b) Write an SQL query that returns the id, city, and orders your results by the number_of_rooms in ascending order.

(a)
```
SELECT *
FROM airbnb_listings;
```
← ; required at end of your queries

(b)
```
SELECT id, city
FROM airbnb_listings
ORDER BY number_of_rooms ASC;
```
← DESC

[order your results by a criteria

SQL Query Examples 2

(a) Get all listings where the number_of_rooms is less than 3.
(b) Get all listings where the number_of_rooms ranges from 3 to 6.

(a)
```
SELECT *
FROM airbnb_listings
WHERE number_of_rooms < 3;
```

(b)
```
SELECT *
FROM airbnb_listings
WHERE number_of_rooms BETWEEN 3 AND 6;
```

first row returned

SQL Query Examples 3

(a) Write a query that returns all listings based in the USA and France.
(b) Return all rows where there are missing values in number_of_rooms.

(a)
```
SELECT *
FROM airbnb_listings
WHERE country in ('USA', 'France');
```

(b)
```
SELECT *
FROM airbnb_listings
WHERE number_of_rooms IS NULL;
```

characters/strings

SQL Query Examples 4

(a) For each country, get the average number of rooms per listing, sorted by ascending order.

(b) Get all the years where there are were more than 100 listings per year.

GROUP BY

AVG

ASC

alias

```
(a) SELECT country, AVG(number_of_rooms) as avg_rooms
    FROM airbnb_listings
    GROUP BY country
    ORDER BY avg_rooms ASC;
```

used after group by

```
HAVING COUNT(id)>100
```

```
(b) SELECT year_listed
    FROM airbnb_listings GROUP BY year_listed
```

SQL Query Join Example

Suppose that you have two tables, orders and customers where customer_id is the primary key of the orders table and the foreign key of the customers table. Write an SQL query that performs an inner join between the tables.

```
SELECT *
FROM orders
INNER JOIN customers ON orders.customer_id =
customers.customer_id;
```

linking based on primary & foreign keys

Time to Level Up with SQLite

There are many SQL dialects that one may use within R to perform high-level data manipulation and dataset creation with SQL such as Postgre SQL and SQLite. For this class, we will be using SQLite to write queries that create and retrieve data with some pre-existing datasets.

My preference

CECS 323

To learn more about advanced SQL concepts for data science or self-reference (indexes, subqueries, query optimization), *The Language of SQL* by Larry Rockoff and Datacamp's SQL for Data Science articles are awesome references to further your knowledge.

Next Time...

Hypothesis Testing in R

Announcements

- Homework 3 is due this Saturday at 11:59pm!
- Midterm Part A Study Guide is posted in Canvas, solutions will be released next week.