

Pro Dev Session



Pro Dev Session

You may want to write this down.

Collaboration is an essential part of being a great teammate and developer. To be a good collaborator, we must master our collaboration tools, namely git and GitHub.

Today we will get more practice with git. We'll be building on this for the next several lessons.



Pro Dev Session

You may want to write this down.

Recall what we've learned so far ...





Check-in Time

- What git command do you use to temporarily save changes that you don't want a permanent record of?
- What git command is used to permanently save files exactly as they are in that moment?
- What git command tracks (or stages) all files in a project?
- What git command is used to create a safe isolated place to work on new features?
- What git command is used to see a list of all previous saved versions?



Git Demo

WATCH & LEARN

Close your laptop. Eyes on my screen. Pay attention.

Pay careful attention to this demo of:

```
git clone
git add -A
git commit -m "<message here>"
git status
git stash
git branch <branchname>
git log
git checkout <branchname or SHA>
```

Pro Dev Session

You may want to write this down.

Today we will get more practice with these commands before we add new ones in the next class.



Git Practice

Work individually but with the support of your classmates to complete all of the goals below.

Goals:

- Create a new repository on GitHub
- Clone the repository to your local machine
- Create a new IntelliJ project with a main method that prints Hello World
- Add, commit, and push your changes to master
- Create a new branch called math and switch to this branch
- Use the `status` command to validate that you are on the correct branch.
- Create an add, subtract, multiply, divide, and exponentiate method.
- Overload at least two of the method.
- Add and commit these changes.
- Return to the master branch. Create a new branch called concat and switch to it.
- Add a method that concatenates multiple strings. Overload this method.
- Use `git pull origin master` to ensure your branch is up to date with the master on GitHub.
- Add, commit, and push your changes using `git push origin concat`.
- On github, create a pull request and merge it.
- Locally, switch to your master branch and use `pull` to update it. With the changes.
- Switch to your math branch. Determine the best way to create a pull request and merge it into master.



**20
minutes!**

Stand Up!



Objectives & Key Outcomes

THE TAKEAWAYS FROM THIS CLASS

By the end of class today, you will be able to:

Better explain
the role of a
database in a
web application



Articulate the
differences
between 1st, 2nd,
and 3rd normal
form



Normalize a
simple
database.



Use SQL to
perform basic
CRUD
operations.



Use joins to
aggregate data
across tables.

Objectives & Key Outcomes

THE TAKEAWAYS FROM THIS CLASS

By the end of class today, you will be able to:



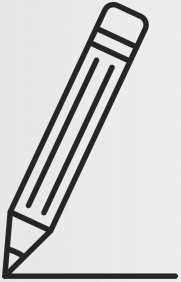
Let's Do This!



Databases

Databases

PERSISTENT DATA SOLUTIONS



Notebooks Ready? It's time for a brief lecture.





Check-in Time

- What HTTP method is used to add new information to our server?



Databases

PERSISTENT DATA SOLUTIONS

We can POST new information to our server, but have you noticed that all that new information goes away when we restart our server?

We need a way to persist data permanently. Something not impacted by power outages or restarts. Something like a giant hard drive.

This is where databases enter the picture.

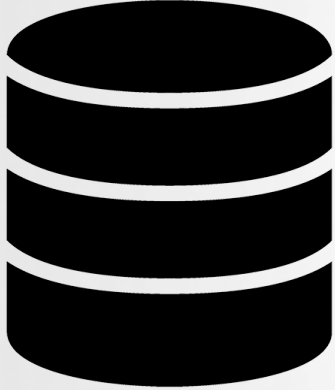
Recall the role of a database...



Web Applications

REVIEWING HOW WEB APPS WORK

Web applications are composed of 3 major components



Database



Server



Client

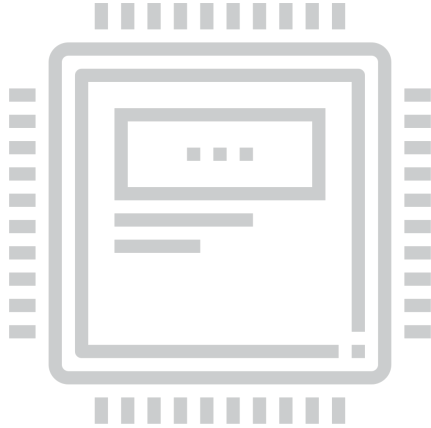
Web Applications

REVIEWING HOW WEB APPS WORK

Clients are the computers rendering the web app. Like your computer. My computer. The computer of that person sitting next to you.



Database



Server



Client

Web Applications

REVIEWING HOW WEB APPS WORK

Servers manage the flow of communication between the client and the database. Servers are how the data from the database gets to page you are viewing.



Database



Server

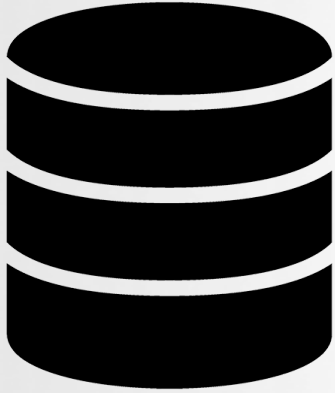


Client

Web Applications

REVIEWING HOW WEB APPS WORK

Databases store data. On a social media site, this might be your username, password, friends, birthday, posts, etc.



Database



Server

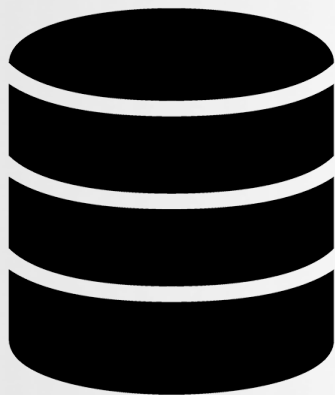


Client

How the Web Works

EXAMPLE WEB APPLICATION

Databases are like complex spreadsheets.



Database

Vickie's data

A curved arrow pointing from the text 'Vickie's data' to the first row of the table.

MOVIE	GENRE	REPEAT	RATIN
Old School	comedy	true	G 4.9
Jurassic Park	action	false	2.7
Bill and Ted	comedy	false	4.8
Mall Cop	comedy	true	5.0
It	horror	false	0.2

Databases

PERSISTENT DATA SOLUTIONS

Databases are essentially really complex spreadsheets where one sheet (or table) can have relationships with another sheet (or table).

Before we dive too far into the theory, let's develop a common vocabulary around databases.



Databases

PERSISTENT DATA SOLUTIONS

Entities are the things we want to track.

A person is an entity.

An order, a shipment, and a payment are all entities.

Attributes tell us about the things we are tracking.

A person has attributes like: Height, weight, age, hair color

An order has attributes like: Customers, costs, placement dates

A tuple is an instance of an entity.

There are x number of people in this room. If we had a student entity, each of you would be a tuple.



Databases

PERSISTENT DATA SOLUTIONS

Theory

The academic/theoretical terms are entities, attributes, and tuples.

Practice

In practice, we call them tables, columns, and rows.

Think of this like an Excel spreadsheet where tables are sheets.

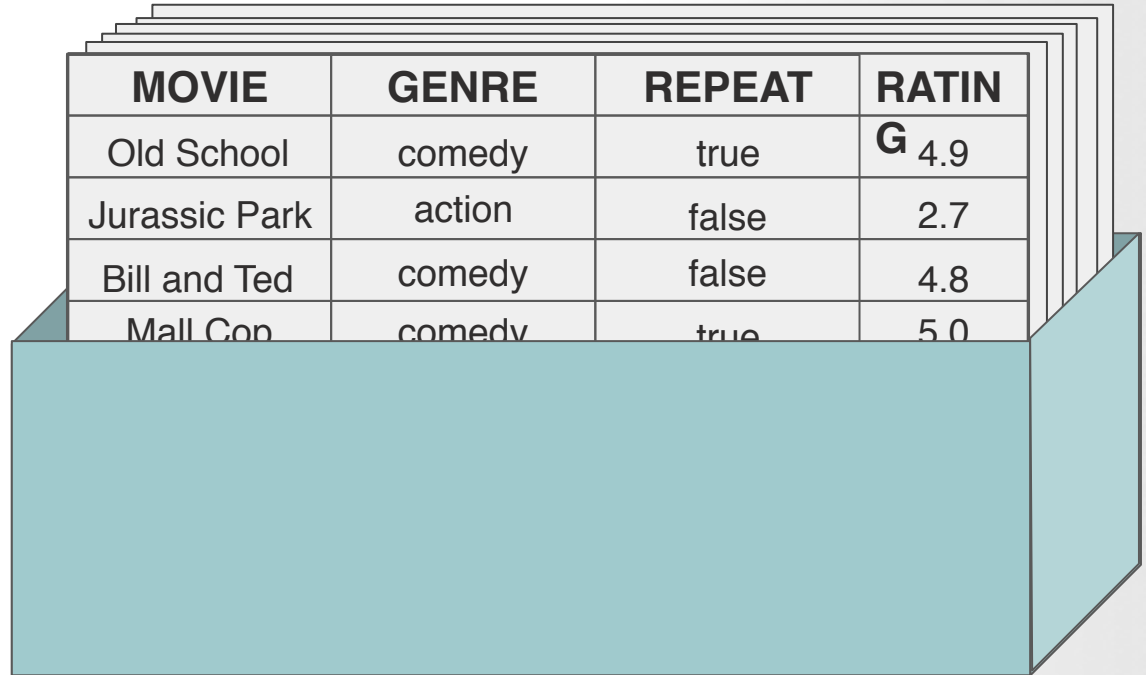


Databases

PERSISTENT DATA SOLUTIONS

A database is kind of like a box where you keep all of your tables. Multiple tables can be stored in a single database.

A project can involve multiple databases.



MOVIE	GENRE	REPEAT	RATIN
Old School	comedy	true	G 4.9
Jurassic Park	action	false	2.7
Bill and Ted	comedy	false	4.8
Mall Cop	comedy	true	5.0

Database Thought Experiment

INDEPENDENT PRACTICE

It's time to fly. Focus. Work hard. Ask for help when you need it.

Work in GROUPS of 4 to complete all of the goals below.

Goals:

- Think about amazon. On a piece of paper design the tables you think you would need to model Amazon's data. Include the columns in each table.
- Begin by determining the entities you will need to track. Consider customers and products to get you started (you need to identify more yourself).
- Once you've listed the entities you need to track, consider what attributes you need to track on each entity.



**20
minutes!**

Databases

PERSISTENT DATA SOLUTIONS

This exercise was tricky but important! In the real world, stakeholders bring us application requirements, but it's our job as data modelers to turn this information into entities/tables!

Let's discuss some common rules for modeling and organizing data.



Databases

PERSISTENT DATA SOLUTIONS

We need to start by identifying our entities, attributes, and relationships.

To begin, we want to track the following information. A website the scale of Amazon would track a lot more, but we want to keep it simple to start.

We need:

- Customer Name
- Customer Address
- Customer Order Info



Databases

PERSISTENT DATA SOLUTIONS

For Amazon our starting entities might be customers and products with the relationship that one customer purchases multiple products.

Our sample Customer table might look like this:

[illegible]

Databases

PERSISTENT DATA SOLUTIONS

Once we have a starting point, we can begin the process of normalizing our data.

1st Normal Form:

1NF has 2 rules:

1. Each row must have a unique identifier.
2. There are no repeating groups of columns.



Databases

PERSISTENT DATA SOLUTIONS

A Note on primary keys:

- Some keys are **natural**. These are attributes that are unique by definition. For example, a social security number, a phone number, or an email address.
- Others are artificially created, like an account number or student id.
- Every table/entity should have a **primary key**.

A primary key is one of these unique values that we choose as the primary identifier of a given row. In the case of a student management system, it would be your student id.



Databases

PERSISTENT DATA SOLUTIONS

1st Normal Form:

Then we eliminate groups columns, like orders. We could have more than or less than 6 orders, so having a column for each doesn't make sense.

id	f_name	l_name	age	birth_date	order
1	tasha	brooke	34	10/22/84	chair
2	tasha	brooke	34	10/22/84	shirt
3	tasha	brooke	34	10/22/84	shoes

Databases

PERSISTENT DATA SOLUTIONS

Once we have achieved 1NF, we can begin the process of normalizing our data to 2NF.

2nd Normal Form:

2NF has 2 rules:

1. The table is in 1NF.
2. All columns are dependent on the table's primary key.

Databases

PERSISTENT DATA SOLUTIONS

2nd Normal Form:

We already know the table is in 1NF. But the columns are not dependent on the primary key. Look tasha brooks is associated with 3 primary keys.

We need to split this into 2 tables so that each column is dependent on the primary key.

id	f_name	l_name	age	birth_date	order
1	tasha	brooke	34	10/22/84	chair
2	tasha	brooke	34	10/22/84	shirt
3	tasha	brooke	34	10/22/84	shoes

Databases

PERSISTENT DATA SOLUTIONS

2nd Normal Form:

The tables are now in 2NF, but we've lost the association between customers and products. We need an intersection table.

id	f_name	l_name	age	birth_date
1	tasha	brooke	34	10/22/84
2	tati	sams	30	12/31/88
3	lily	banks	20	03/15/99

CUSTOMER
TABLE

id	product	price
1	chair	31.22
2	shirt	5.45
3	shoe	72.00

PRODUCT TABLE

Databases

PERSISTENT DATA SOLUTIONS

2nd Normal Form:

This 3rd table associates customers with products and represents the orders

id	f_name	l_name	age	birth_date
1	tasha	brooke	34	10/22/84
2	tati	sams	30	12/31/88
3	lily	banks	20	03/15/99

CUSTOMER
TABLE

id	product	price
1	chair	31.22
2	shirt	5.45
3	shoe	72.00

PRODUCT TABLE

id	cust_id	prod_id
1	1	3
2	1	1
3	1	2

ORDER TABLE

Databases

PERSISTENT DATA SOLUTIONS

Once we have achieved 2NF, we can begin the process of normalizing our data to 3NF.

3rd Normal Form:

3NF has 2 rules:

1. The table is in 2NF.
2. No columns are dependent on each other.



Databases

PERSISTENT DATA SOLUTIONS

3rd Normal Form:

Note that age is dependent on birth_date. This is a dependency between two columns. In this case, we can simply remove age from our table and calculate it as needed.

id	f_name	l_name	age	birth_date
1	tasha	brooke	34	10/22/84
2	tati	sams	30	12/31/88
3	lily	banks	20	03/15/99

CUSTOMER
TABLE

id	product	price
1	chair	31.22
2	shirt	5.45
3	shoe	72.00

PRODUCT TABLE

id	cust_id	prod_id
1	1	3
2	1	1
3	1	2

ORDER TABLE

Databases

PERSISTENT DATA SOLUTIONS

3rd Normal Form:

Note that age is dependent on birth_date. This is a dependency between two columns. In this case, we can simply remove age from our table and calculate it as needed.

id	f_name	l_name	birth_date
1	tasha	brooke	10/22/84
2	tati	sams	12/31/88
3	lily	banks	03/15/99

CUSTOMER
TABLE

id	product	price
1	chair	31.22
2	shirt	5.45
3	shoe	72.00

PRODUCT TABLE

id	cust_id	prod_id
1	1	3
2	1	1
3	1	2

ORDER TABLE

Databases

PERSISTENT DATA SOLUTIONS

3rd Normal Form:

Note that in this table we have maker which depends on game because every game has only 1 maker and it may change. This may not seem problematic but what happens when we enter Scrabble into the database. It is also made by Hasbro. We would enter the same maker twice. A simple typo in the maker name could result in major data inconsistencies. Maker should be moved to a new table and linked by its id.

id	game	maker
1	Monopoly	Hasbro
2	Pictionary	Mattel
3	Pandemic	Z-MAN

Database Thought Experiment

INDEPENDENT PRACTICE

It's time to fly. Focus. Work hard. Ask for help when you need it.

Work in GROUPS of 4 to complete all of the goals below.

Goals:

- Design a system for tracking students and the courses they are taking.
- You should have tables for student information, course information, and a table for enrollments, which links the previous two
- Remember to keep your tables in third normal form



**15
minutes!**

Objectives & Key Outcomes

THE TAKEAWAYS FROM THIS CLASS

By the end of class today, you will be able to:

Better explain
the role of a
database in a
web application



Normalize a
simple
database.



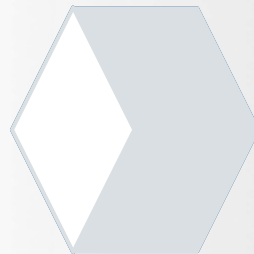
Articulate the
differences
between 1st, 2nd,
and 3rd normal
form.



Use SQL to
perform basic
CRUD
operations.



Use joins to
aggregate data
across tables.





Stay Seated & Take 3 Deep Breaths.

RELAX.

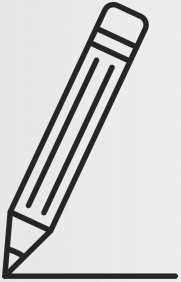
Now take a short walk. Clear your head. After a few minutes break, quickly review your notes.
We'll start back shortly.



SQL

SQL

STRUCTURED QUERY LANGUAGE



Notebooks Ready? It's time for a brief lecture.



SQL


STRUCTURED QUERY LANGUAGE

We are about to learn a new language - SQL. SQL is a very popular database language. This language allows us to interact with the database directly, which we'll be doing today.

Going forward we will be using some Java tools (which we'll see at the end of class) to do this work for us, but it's important to know SQL and understand what these Java tools are doing behind the scenes.

Again, **we will not be writing SQL after today**, but we will be doing all of these same actions using a Java tool that handles the SQL for us.

SQL is very important. The same way a mechanic needs to know how to diagnose car issues even though diagnostics are predominantly handled by an onboard computer these days, a software engineer must know SQL even though Java has tools that will typically handle the heavy lifting.



SQL

STRUCTURED QUERY LANGUAGE

To create a database, connect to the MySQL server in MySQL Workbench. Then we can execute commands.

```
CREATE DATABASE student_db;
```

SQL

CODE-A-LONG

Open your laptop. Code with me. Don't jump ahead.

Follow along with me as I complete this activity.

Goals:

- Using workbench create a table representing the students in this class

SQL

STRUCTURED QUERY LANGUAGE

```
USE student_db;
```

```
INSERT INTO students (student_name, student_age, is_enrolled,  
date_enrolled) VALUES ("George Rodriguez", 22, true, "2019-04-07");
```


SQL

INDEPENDENT PRACTICE

It's time to fly. Focus. Work hard. Ask for help when you need it.

Work in PAIRS to complete all of the goals below.

Goals:

- Using workbench and the previous slide, add all the students in your row to the table.



5 minutes!

SQL

STRUCTURED QUERY LANGUAGE

```
SELECT * FROM students;
```



SQL

CODE-A-LONG

Open your laptop. Code with me. Don't jump ahead.

Follow along with me as I complete this activity.

Goals:

- Using workbench, update and delete records in a table

SQL

INDEPENDENT PRACTICE

It's time to fly. Focus. Work hard. Ask for help when you need it.

Work in PAIRS to complete all of the goals below.

Goals:

- Create a new database called restaurant_db
- Create a new table called `restaurants` with the following columns:
 - id (primary key)
 - Restaurant_name
 - Restaurant_type
 - Permanently_closed
- Insert at least 5 rows of data to the new table.
- Update the 3rd restaurant to be named “Jimmy’s Chicken and Biscuits
- Delete the 2nd restaurant you added.



**10
minutes!**

Objectives & Key Outcomes

THE TAKEAWAYS FROM THIS CLASS

By the end of class today, you will be able to:

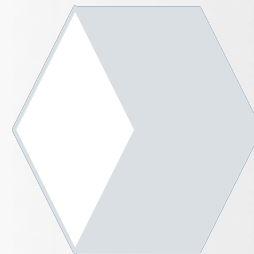
Better explain
the role of a
database in a
web application



Normalize a
simple
database.



Use SQL to
perform basic
CRUD
operations.



Use joins to
aggregate data
across tables.

Articulate the
differences
between 1st, 2nd,
and 3rd normal
form.



lunch.

Joins



Joins

AGGREGATING DATA ACROSS TABLES



Notebooks Ready? It's time for a brief lecture.



Joins

AGGREGATING DATA ACROSS TABLES

Having relational data in a database is of limited utility if we can't aggregate that data in some way. For this purpose, we use **joins**.

Joins take data from multiple tables and join them into one cohesive data set.

The main three joins we're going to take a look at are:

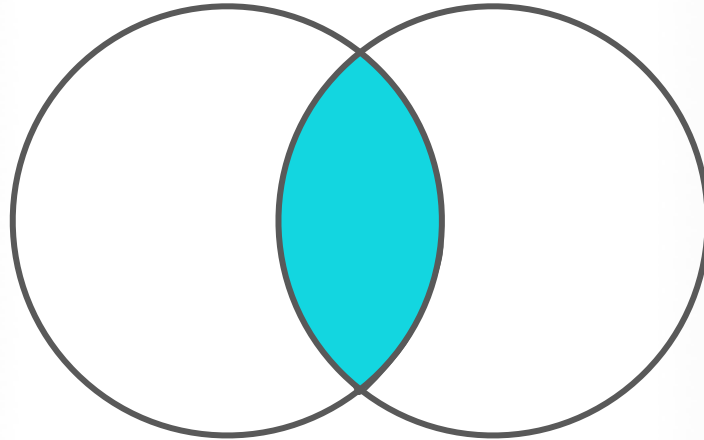
- **Inner**
 - Returns only rows that intersect in Table A and Table B
- **Left**
 - Returns all of Table A along with any intersecting rows from Table B
- **Right**
 - Returns all of Table B along with any intersecting rows from Table A

Note: In practice, right joins are rarely used, as they are functionally equivalent to left joins

Joins

AGGREGATING DATA ACROSS TABLES

Inner Join

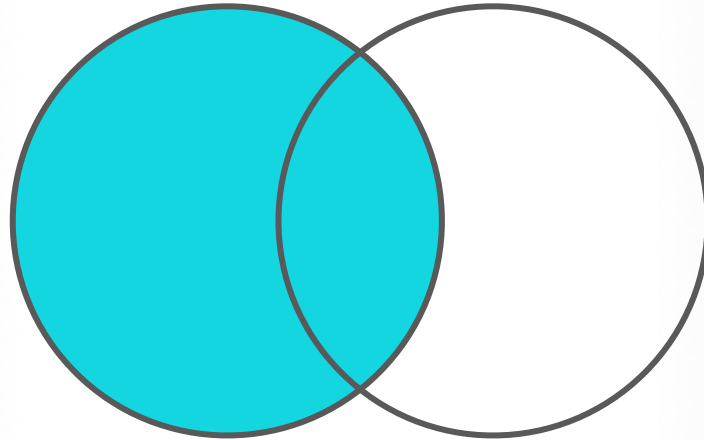


```
SELECT <select_list> FROM Table_A INNER JOIN Table_B ON A.Key = B.Key
```

Joins

AGGREGATING DATA ACROSS TABLES

Left Join

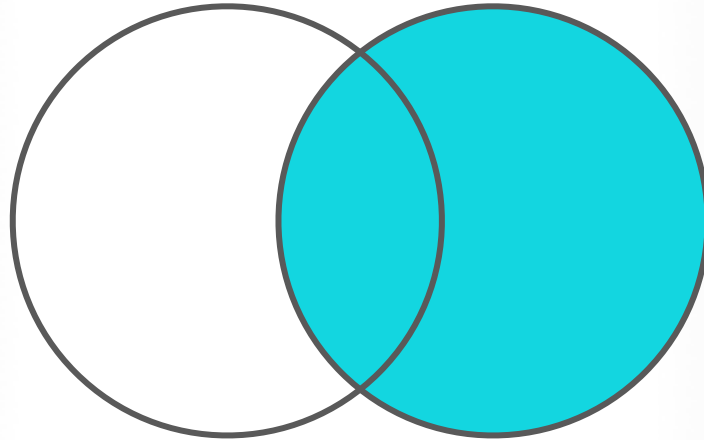


```
SELECT <select_list> FROM Table_A LEFT JOIN Table_B ON A.Key = B.Key
```

Joins

AGGREGATING DATA ACROSS TABLES

Right Join



```
SELECT <select_list> FROM Table_A RIGHT JOIN Table_B ON A.Key = B.Key
```

Joins

WATCH & LEARN

Close your laptop. Eyes on my screen. Pay attention.

```
SELECT
    customers.first_name,
    customers.last_name,
    orders.order_date,
    orders.order_amount
FROM customers LEFT JOIN orders
    ON customers.customer_id = orders.customer_id;
```

SQL

CODE-A-LONG

Open your laptop. Code with me. Don't jump ahead.

Work in PAIRS to complete all of the goals below.

Goals:

- Run the provided schema and seeds in MySQL Workbench to create the tables and add data to them.
- List the **book title**, **author's first name**, and **author's last name** only for entries that have corresponding data in both tables



**15
minutes!**



Stay Seated & Take 3 Deep Breaths.

RELAX.

Now take a short walk. Clear your head. After a few minutes break, quickly review your notes.
We'll start back shortly.



Objectives & Key Outcomes

THE TAKEAWAYS FROM THIS CLASS

By the end of class today, you will be able to:

Better explain
the role of a
database in a
web application



Articulate the
differences
between 1st, 2nd,
and 3rd normal
form



Normalize a
simple
database.



Use SQL to
perform basic
CRUD
operations.



Use joins to
aggregate data
across tables.

Lab Time



LabTime

INDEPENDENT PRACTICE

It's time to fly. Focus. Work hard. Ask for help when you need it.

Work together but INDEPENDENTLY write your own code to complete all of the goals below.

Goals:

- Complete all the katas listed in the activity file.

If this is tough, great! You're getting practice.

If this is easy, great! Help your buddies.


We'll be circulating to provide individual help where it's needed.



**45
minutes!**

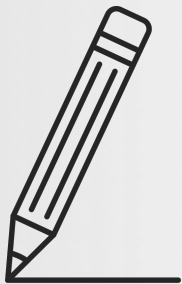
Sneak Peek

Note that next class will start with the sneak peek topic. You are NOT expected to master this today.



JPA

JAVA PERSISTENCE API



Notebooks Ready? It's time for a brief lecture.

JPA

JAVA PERSISTENCE API

JPA is the **Java Persistence API**

Using JPA, we can set up Data Transfer Objects, or **DTOs**, to map database records to Java objects.

This is accomplished through the use of **annotations**.



JPA

WATCH & LEARN

Close your laptop. Eyes on my screen. Pay attention.

```
@Entity
@JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})
@Table(name="customer")
public class Customer {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer id;
    private String firstName;
    private String lastName;
    private String company;
    private String phone;

    @OneToMany(mappedBy = "customerId", cascade = CascadeType.ALL, fetch =
FetchType.EAGER)
    private Set<Note> notes;
    // Getters and  setters omitted for brevity
}
```

Wrap Up

Module 4 Lesson 1

HOMework

You don't have to submit your nightly homework, but you are expected to complete it.

For homework tonight, finish the katas.

To assist in completing the katas, read this article on the WHERE clause in SQL statements.

<http://www.sqltutorial.org/sql-where/>



Daily Assessment Today

You may leave after a staff member approves your assessment.



Daily Assessment

Work INDIVIDUALLY to complete all of the goals below.

Goals:

- Complete the CAR LOT portion of the katas if you haven't already.