Data Modeling

Data Boot Camp

Lesson 9.3

Class Objectives

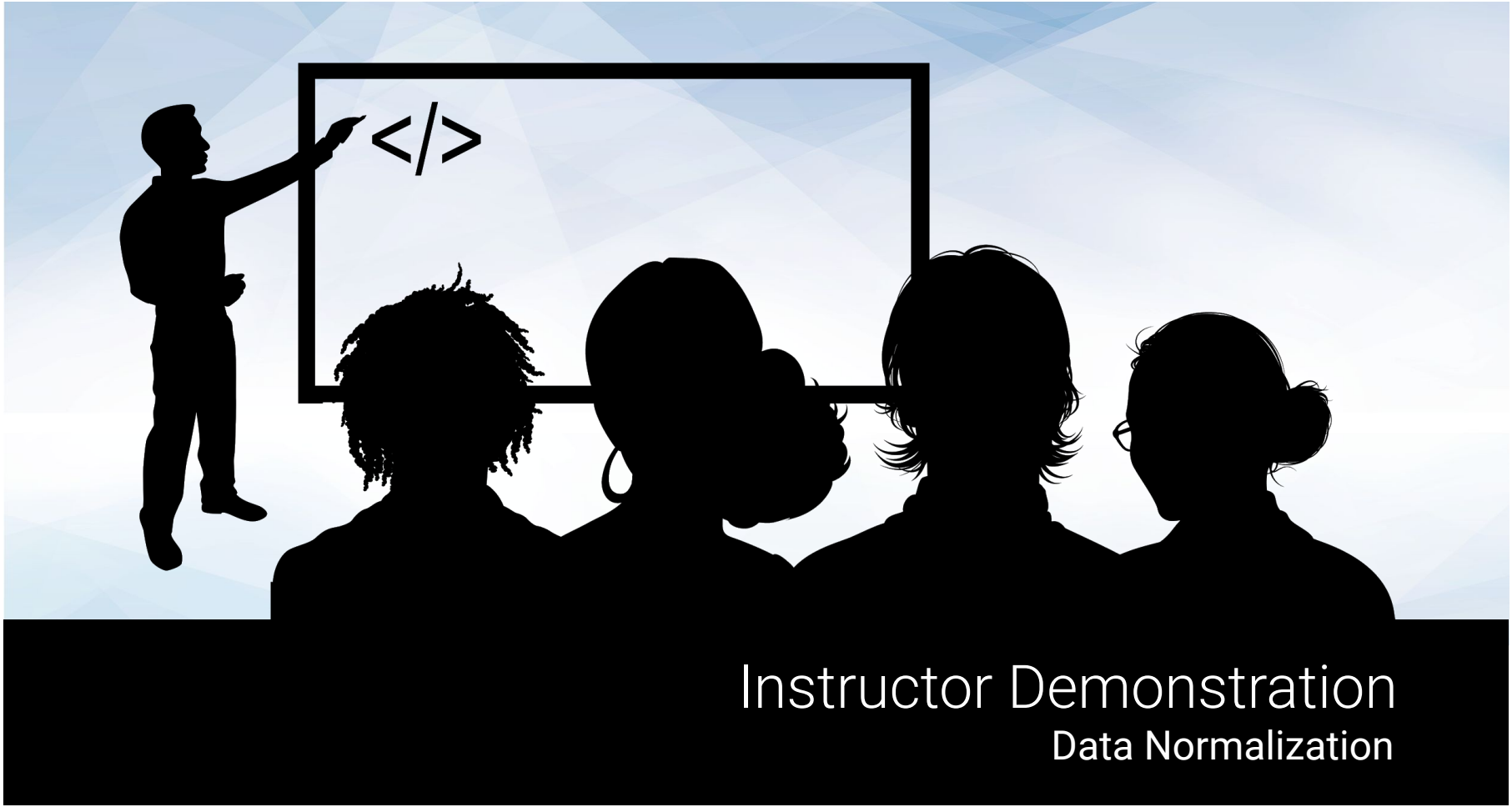
By the end of today's class you will be able to:

Apply data modeling techniques to database design.

Normalize data.

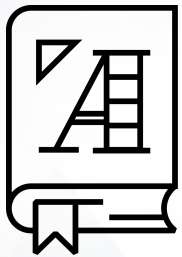
Identify data relationships.

Create visual representations of a database through entity relationship diagrams.



Instructor Demonstration

Data Normalization



Data normalization is the process of restructuring data to a set of defined "normal forms."

The process of data normalization **eliminates data redundancy and inconsistencies**.

Data Normalization

Three main forms of normalization

- Process of restructure data to a set of “normal forms.”
- Reduce and eliminate data redundancy and inconsistencies
- Three most common forms:
  - **First normal form (1NF)**
  - **Second normal form (2NF)**
  - **Third normal form (3NF)**
- There are even more levels!

Data Normalization

Each field in a table row should contain a single value

Each row is unique

- Rows can have a fields that repeat
- But whole rows do not fully match

Raw Data

family	children
Smiths	Chris, Abby, Susy
Jones	Steve, Mary, Dillion

Normalization

First Normal Form

family	children
Smiths	Abby
Smiths	Chris
Smiths	Susy
Jones	Dillon
Jones	Mary
Jones	Steve

Data Normalization (2NF)

- Be in First Normal Form
- Single Column Primary Key
  - Primary Key
  - Identifies the table and row uniquely
- Generally there could be a need to create a new table

Data in 1NF

family	children
Smiths	Abby
Smiths	Susy
Jones	Mary
Smiths	Chris
Jones	Dillion
Jones	Mary

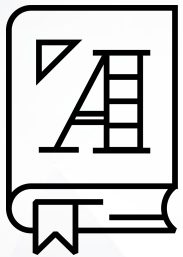
2NF Normalization

Family Table

family_id	family
1	Smiths
2	Jones

Child Table

family_id	children
1	Chris
1	Abby
1	Susy
2	Steve
2	Mary
2	Dillion



Transitive Dependency is the reliance of a column’s value on another column through a third column.

Data Normalization

Transitive Dependency

- Transitive
  - If X>Y and Y>Z then X>Z.
- Dependence
  - One value relies on another.
  - City relies on ZIP code; age depends on birthday.
- For example:
  - Say you have three columns: `StoreName`, OwnerAddress, `OwnerName`.
  - OwnerName and OwnerAddress rely on the the `StoreName`.
  - `OwnerAddress` also relies on the `OwnerName`.
  - So `OwnerAddress` relies on the `StoreName` via the `OwnerName`.

Data Normalization

First Normal Form (1NF) to Second Normal Form (2NF)

- Be in First Normal Form
- Single Column Primary Key
  - Primary Key
  - Identifies the table and row uniquely
- Generally there could be a need to create a new table

owner_name	address	store_name
Marshall	123, Fake St.	Soups and Stuff
Susan	44, New Drive	Sink Emporium
Susan	99, Old Lane	Tasty Burgers

2NF Normalization

store_id	owner_name	address	store_name	type
1	Marshall	123, Fake St.	Soups and Stuff	Food
2	Susan	44, New Drive	Sink Emporium	Appliance
2	Susan	99, Old Lane	Tasty Burgers	Food

Data Normalization

Second Normal Form (2NF) to Third Normal Form (NF3)

- Must be in Second Normal Form
- Contain non-transitively dependent columns

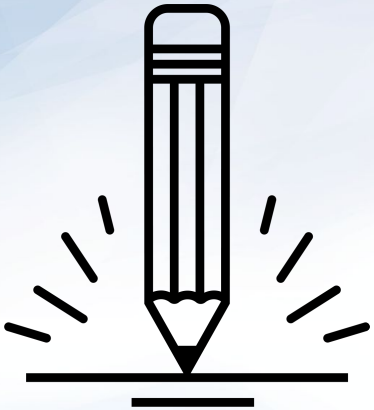
store_id	owner_name	address	store_name	type
1	Marshall	123, Fake St.	Soups and Stuff	Food
2	Susan	44, New Drive	Sink Emporium	Appliance
2	Susan	99, Old Lane	Tasty Burgers	Food

3NF Normalization

store_id	address	store_name	type_id
1	123, Fake St.	Soups and Stuff	1
2	44, New Drive	Sink Emporium	2
2	99, Old Lane	Tasty Burgers	1

store_id	owner_name
1	Marshall
2	Susan
2	Susan

type_id	type
1	Food
2	Appliance



Activity: Pet Normalizer

In this activity, you will practice data normalization skills using the provided data.

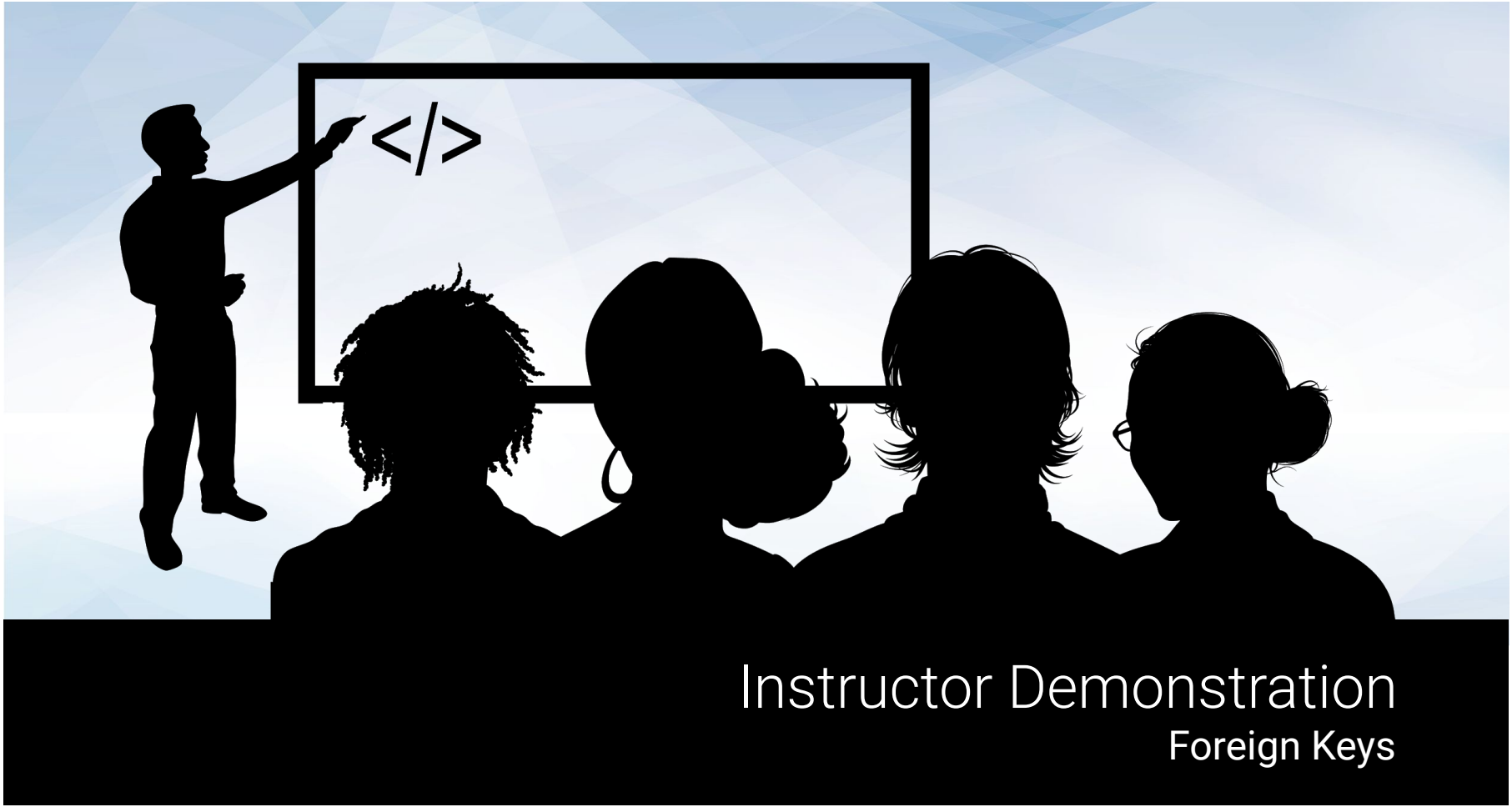
Suggested Time:
15 Minutes

Activity: Pet Normalizer

- In pgAdmin, create a new database called `pets_db`.
- Use Excel to get the data into first normal form (1NF).
- Using the normalized CSV, create the following tables with continued normalized practices:
  - a table for owners that takes an ID and the owner's name
  - a table for pet names that takes two IDs, the pet's name, and the pet's type.
- Using the CSV file as guide, insert the data into respective tables.

- **Hint**: Be sure that each table has a unique primary key. 
- **Bonus:**
  - Create a service table that displays the different types of services that are offered.
  - Create a pet_names_updated table that takes an ID that will connect to the services table.
  - Join all three tables.

Time’s Up! Let’s Review.



Instructor Demonstration

Foreign Keys

Foreign Keys

Foreign Keys reference the primary key of another table.

- Can have a different name
- Do not need to be unique

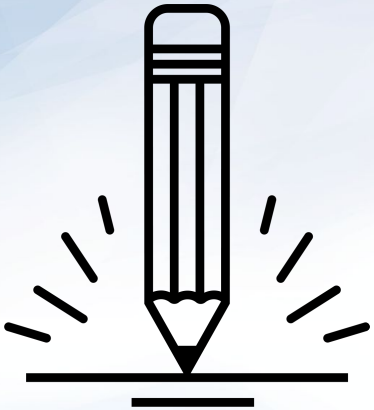
Primary Key

family_id	family
1	Smiths
2	Jones

Primary Key

Foreign Key

child_id	family_id	children
11	1	Chris
22	1	Abby
33	1	Susy
44	2	Steve
55	2	Mary
66	2	Dillion



Activity: Foreign Keys

In this activity, you will create and populate two new tables with foreign keys that reference existing data.

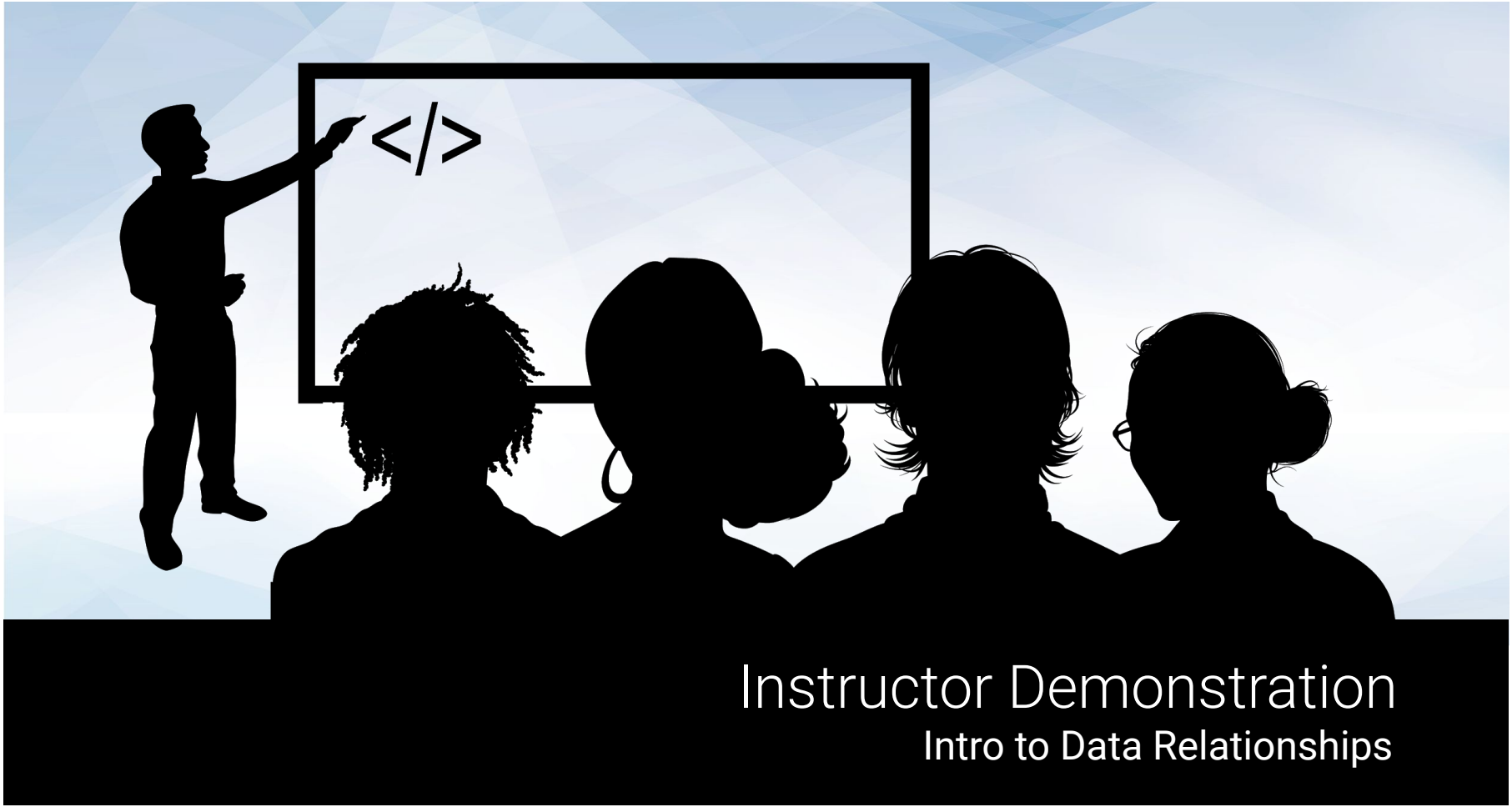
Suggested Time:
15 Minutes

Activity: Foreign Keys

- Create a `customer` table with a customer first name and customer last name.
- Create a `customer_email` table with a foreign key that references a field in the original `customer` table.
- Populate the `customer_email` table with emails.
- Create a `customer_phone` table with a foreign key that references a field in the original customer table.
- Populate the `customer_phone` table with phone numbers.
- Test foreign keys by writing a query to insert data in the `customer_phone` table that does not have a reference ID in the `customer` table.
- Join all three tables.
- **Hint**: Think about how you can select certain columns in a table. Use those columns as a reference to insert data into a table. 

***Note**: Make sure all tables have primary keys that increment with each new row of data.*

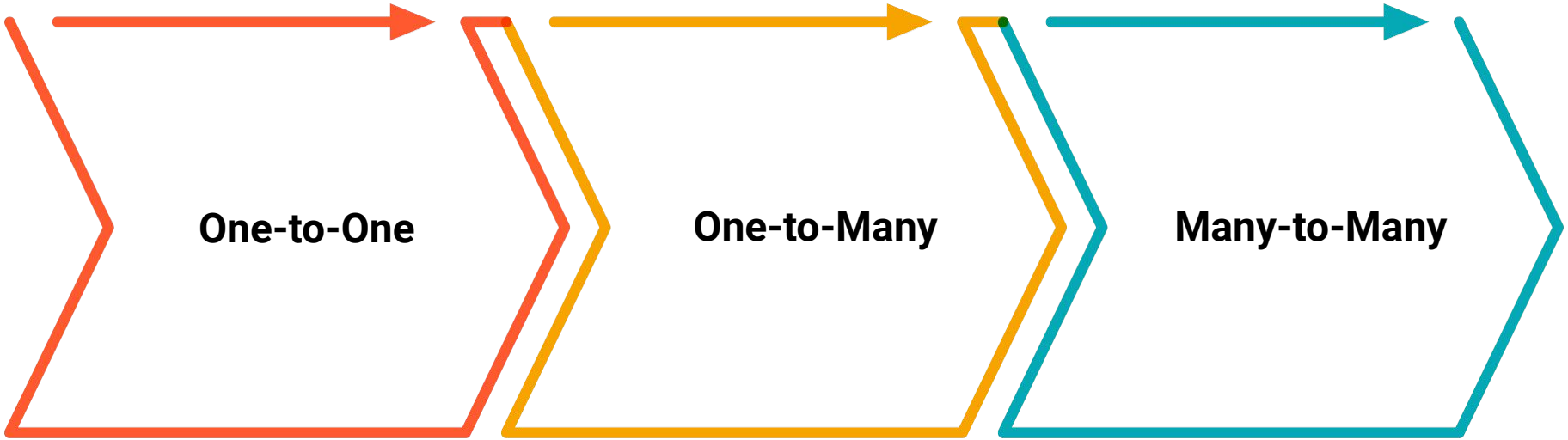
Time’s Up! Let’s Review.



Instructor Demonstration

Intro to Data Relationships

Intro to Data Relationships



Intro to Data Relationships

One-to-One Relationship

ID	Name	Social Security
1	Homer	111111111
2	Marge	222222222
3	Lisa	333333333
4	Bart	444444444
5	Maggie	555555555

Each item in one column is linked to only one other item from the other column.

Here, each person in the Simpsons family can have only one social security number.

Each social security number can be assigned only to one person.

Intro to Data Relationships

One-to-Many Relationship

ID	Address
11	742 Evergreen Terrace
12	221B Baker Street

ID	Name	Social Security	AddressID
1	Homer	111111111	11
2	Marge	222222222	11
3	Lisa	333333333	11
4	Bart	444444444	11
5	Maggie	555555555	11
6	Sherlock	112233445	12
7	Watson	223344556	12

- Each address can be associated with multiple people.
- Each person has an address.
- The two tables, joined, would look like this.

Intro to Data Relationships

Many-to-Many Relationship

ID	Child		ID	Parent
1	Bart		11	Homer
2	Lisa		12	Marge
3	Maggie			

- Each child can have more than one parent.
- Each parent can have more than one child.

Intro to Data Relationships

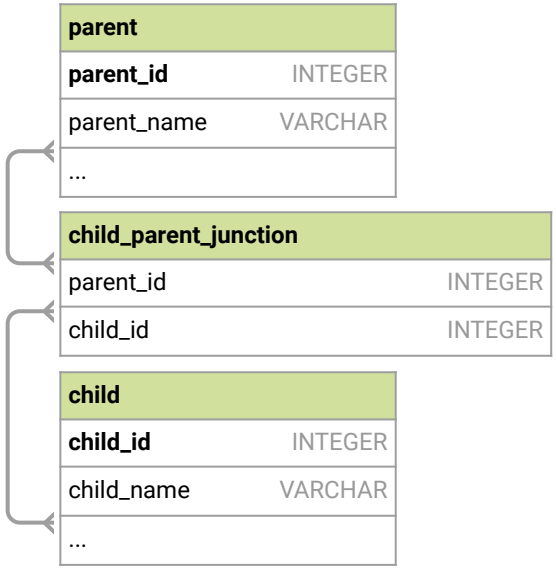
Many-to-Many Relationship

ChildID	Child	ParentID	Parent
1	Bart	11	Homer
1	Bart	12	Marge
2	Lisa	11	Homer
2	Lisa	12	Marge
3	Maggie	11	Homer
3	Maggie	12	Marge

- Each child can have more than one parent.
- Each parent can have more than one child.
- The two tables are joined in a junction table.

Intro to Data Relationships

Junction Table

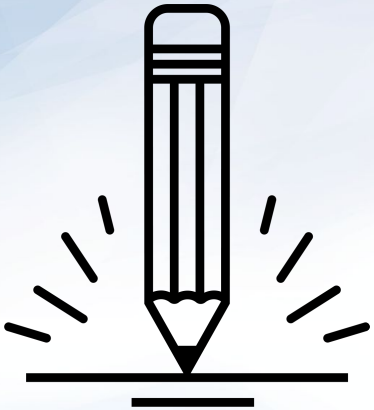


	parent_id integer	child_id integer
1	11	1
2	11	2
3	11	3
4	12	1
5	12	2
6	12	3

Join child and parent table to junction table

	parent_name character varying (255)	child_name character varying (255)
1	Homer	Bart
2	Homer	Lisa
3	Homer	Maggie
4	Marge	Bart
5	Marge	Lisa
6	Marge	Maggie

The Junction table contains many parent_id's and many child_id's



Activity: Data Relationships

In this activity, you will create table schemata for students and available courses, and then create a junction table to display all courses taken by students.

Suggested Time:
15 Minutes

Activity: Data Relationships

You are the database consultant at a new university. Your job is to design a database model for the registrar. The database will keep track of information on students, courses offered by the university, and the courses each student has taken.

- Create a `students` table that keeps track of the following:
  - Unique ID number of each student
  - Last and first names of each student
- Create a `courses` table that keeps track of the following:
  - Unique ID number of each course
  - Name of each course
- Create a `student_courses_junction` that keeps track of the following:
  - All courses that have been taken by each student
  - Term in which a course was taken by a student (spring or fall)
- Which data model is appropriate here: one to one, one to many, or many to many?
- **Bonus:**
  - If time allows, join and query the tables to get all data on the students.

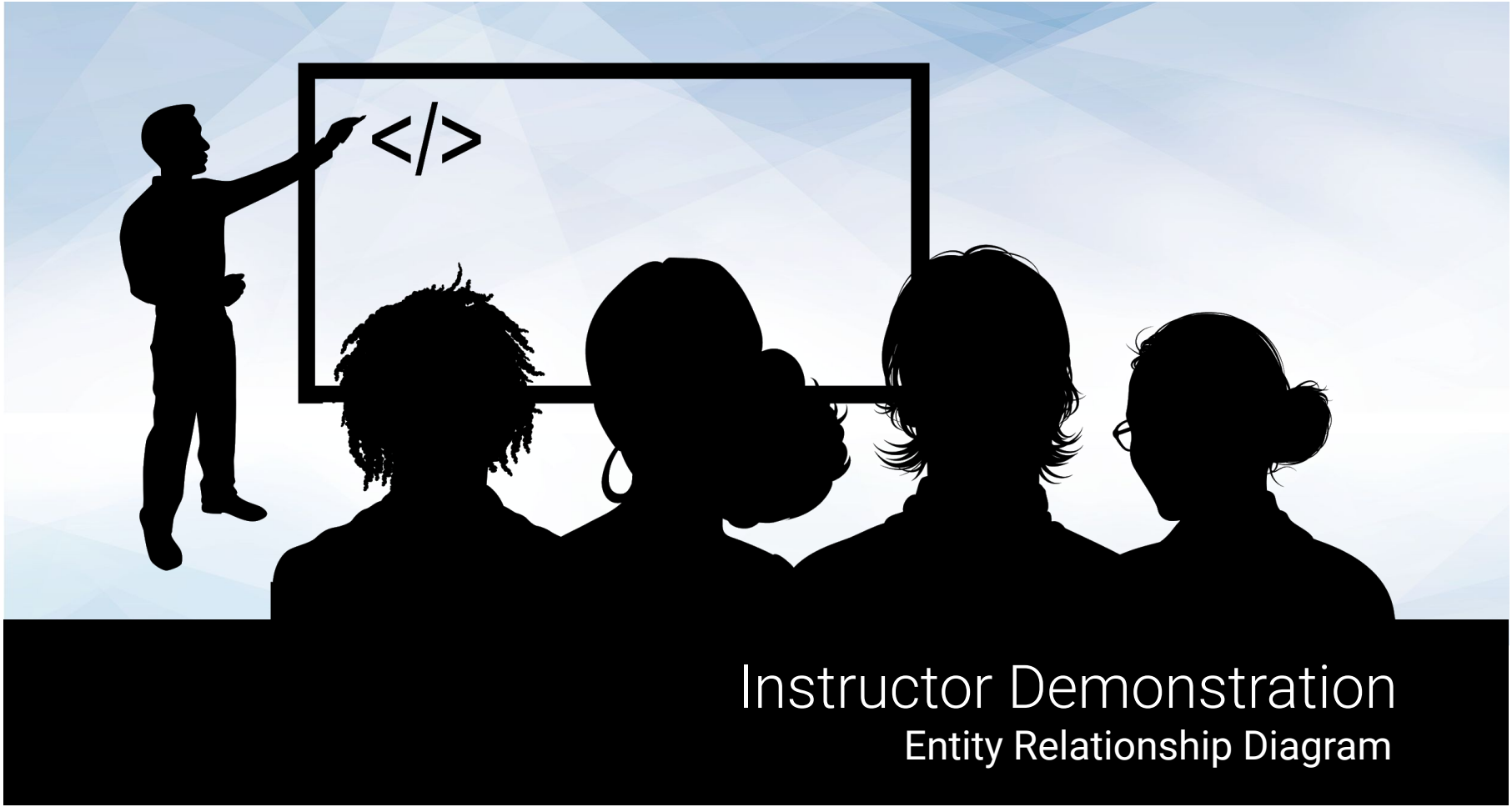
Time’s Up! Let’s Review.

Take a Break!

Countdown timer

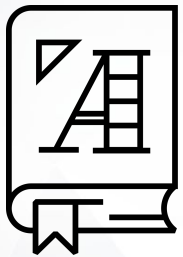
40:00

(with alarm)



Instructor Demonstration

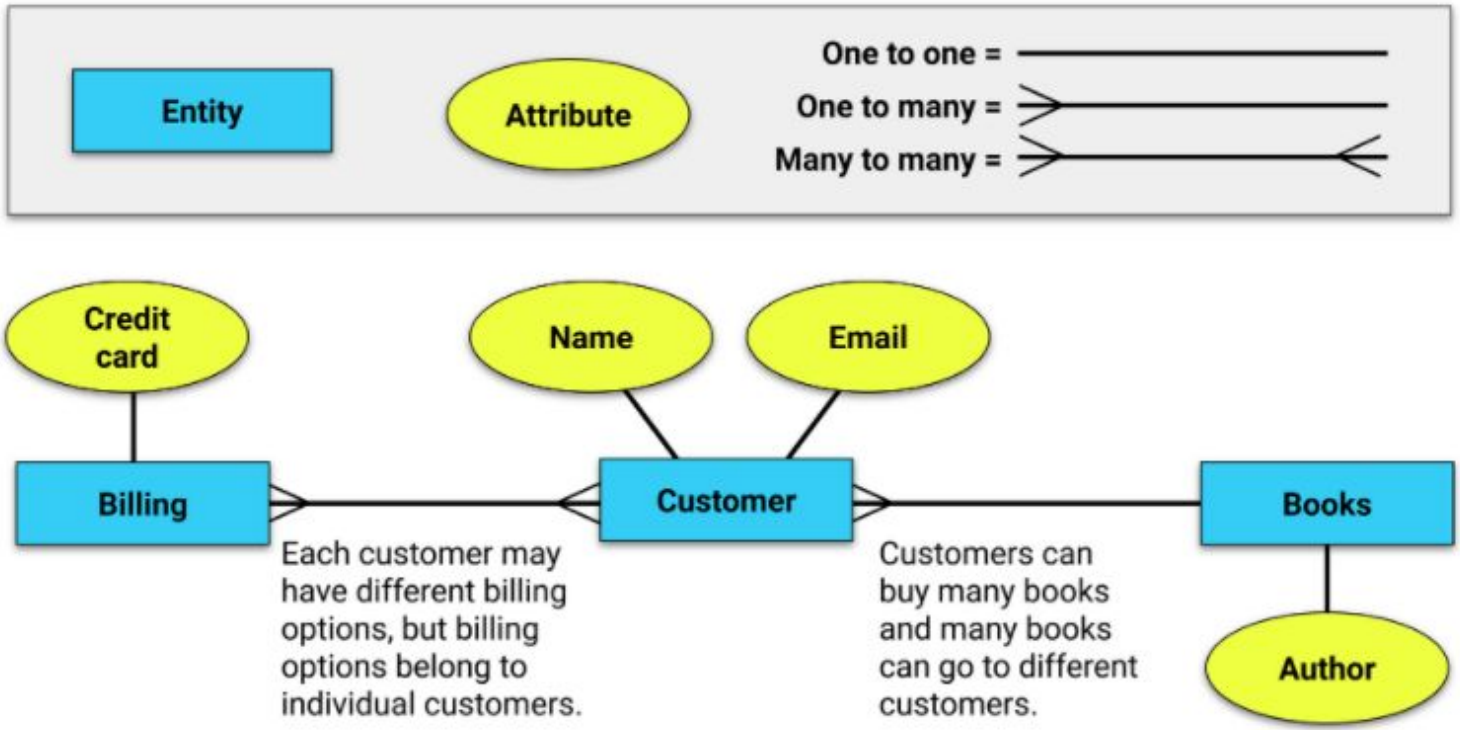
Entity Relationship Diagram



An **entity relationship diagram**, or **ERD**, is a visual representation of entity relationships within a database.

Entity Relationship Diagram

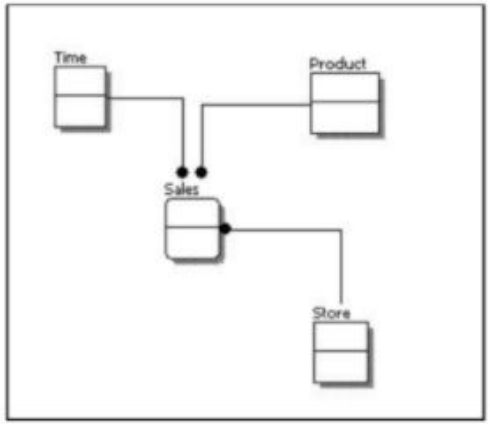
ERD use the following notation to create the models.



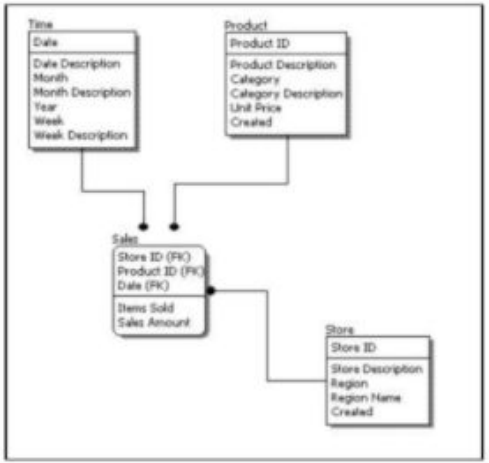
Entity Relationship Diagram

Three Types of ERDs or Data Models

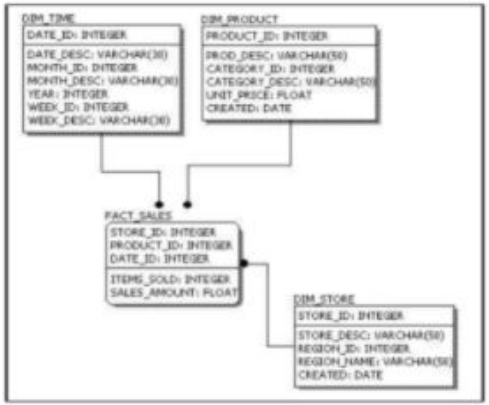
Conceptual Model Design



Logical Model Design



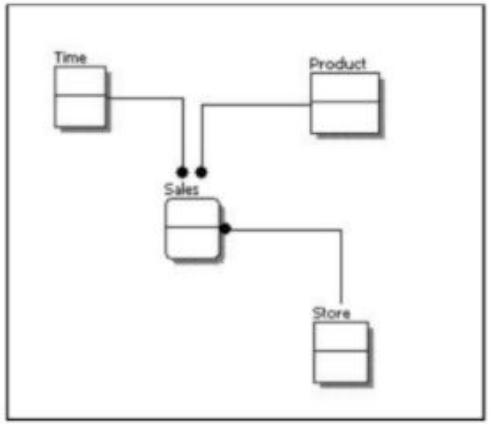
Physical Model Design



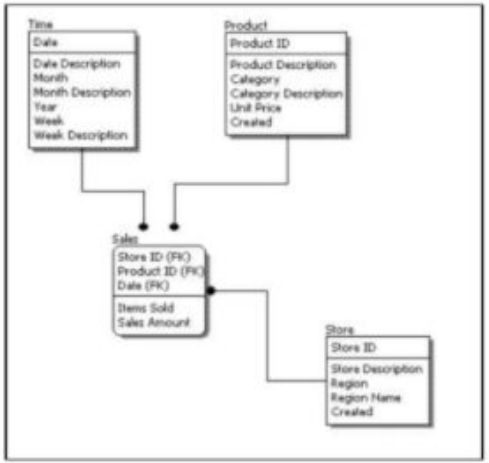
Entity Relationship Diagram

Three Types of ERDs or Data Models

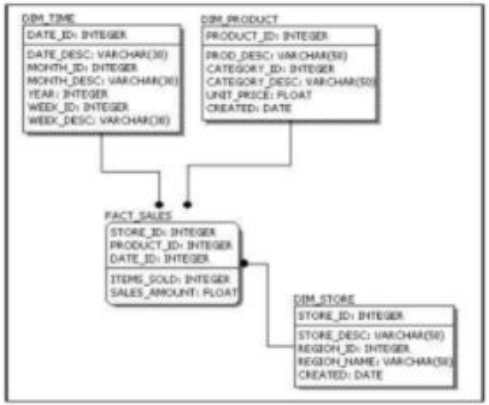
Conceptual Model Design



Logical Model Design

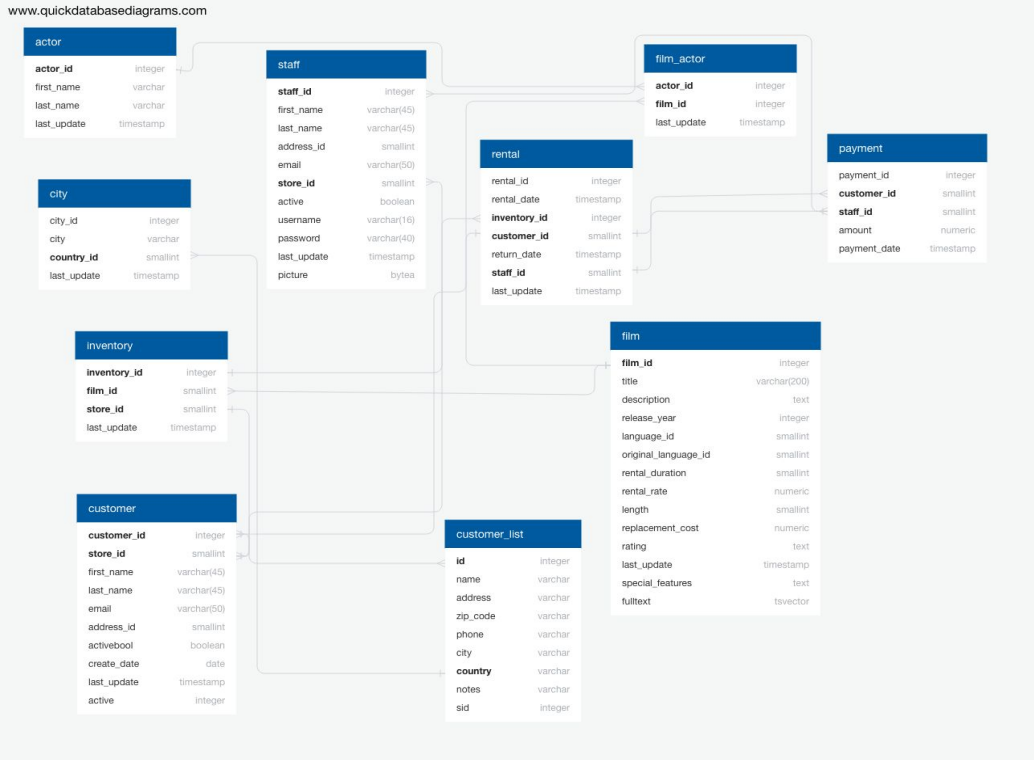


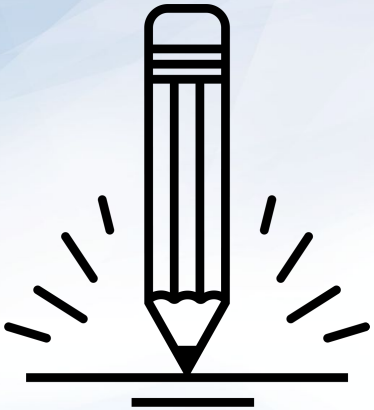
Physical Model Design



Entity Relationship Diagram

Entities, their data types, and relationships are all illustrated in the diagram





Activity: Designing an ERD, Part I

In this activity, you will create a conceptual **ERD** for a gym owner.

Suggested Time:
15 Minutes

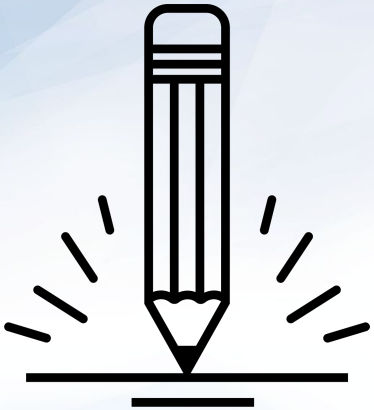
Activity: Designing an ERD, Part I

In this activity, you will work with a partner given the following scenario:

You are meeting with a gym owner who wants to organize his data in a database. Create a conceptual ERD for the gym owner.

- Create a conceptual ERD by determining the entities that will be present in the database, along with their attributes. Be sure to include the following: trainers, members, and gym as well as one more entity that you think is necessary.
- Create a diagram using the Quick Database Diagrams tool.
- When you are satisfied with the conceptual diagram, update it to a logical ERD by including column data types and primary keys.
- Update your existing diagram to reflect the changes.
- Hint:
    - Check your slack for the documentation for more in-depth explanations of entity relationship diagrams.

Time’s Up! Let’s Review.



Activity: Designing an ERD, Part II

In this activity, you and and your partner will continue designing an entity relationship diagram for the gym by transitioning your logical **ERD** created in the previous activity to a physical **ERD.**

Suggested Time:
15 Minutes

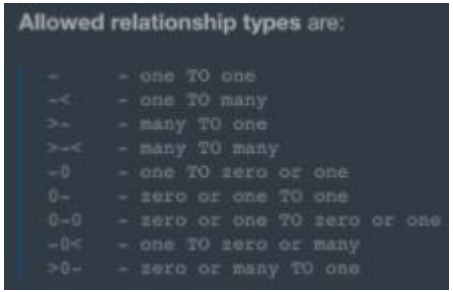
Activity: Designing an ERD, Part II

- Using the starter code provided, return to Quick Database Diagrams and transition your logical ERD to a physical ERD by creating the relationships between tables.
- When you are satisfied with your ERD, write a corresponding schema file containing your `CREATE TABLE` statements
- In pgAdmin, connect to your server and create a new database named `gym`. Then open a query tool.
- Paste the code from your schema file in pgAdmin, and then execute the code.

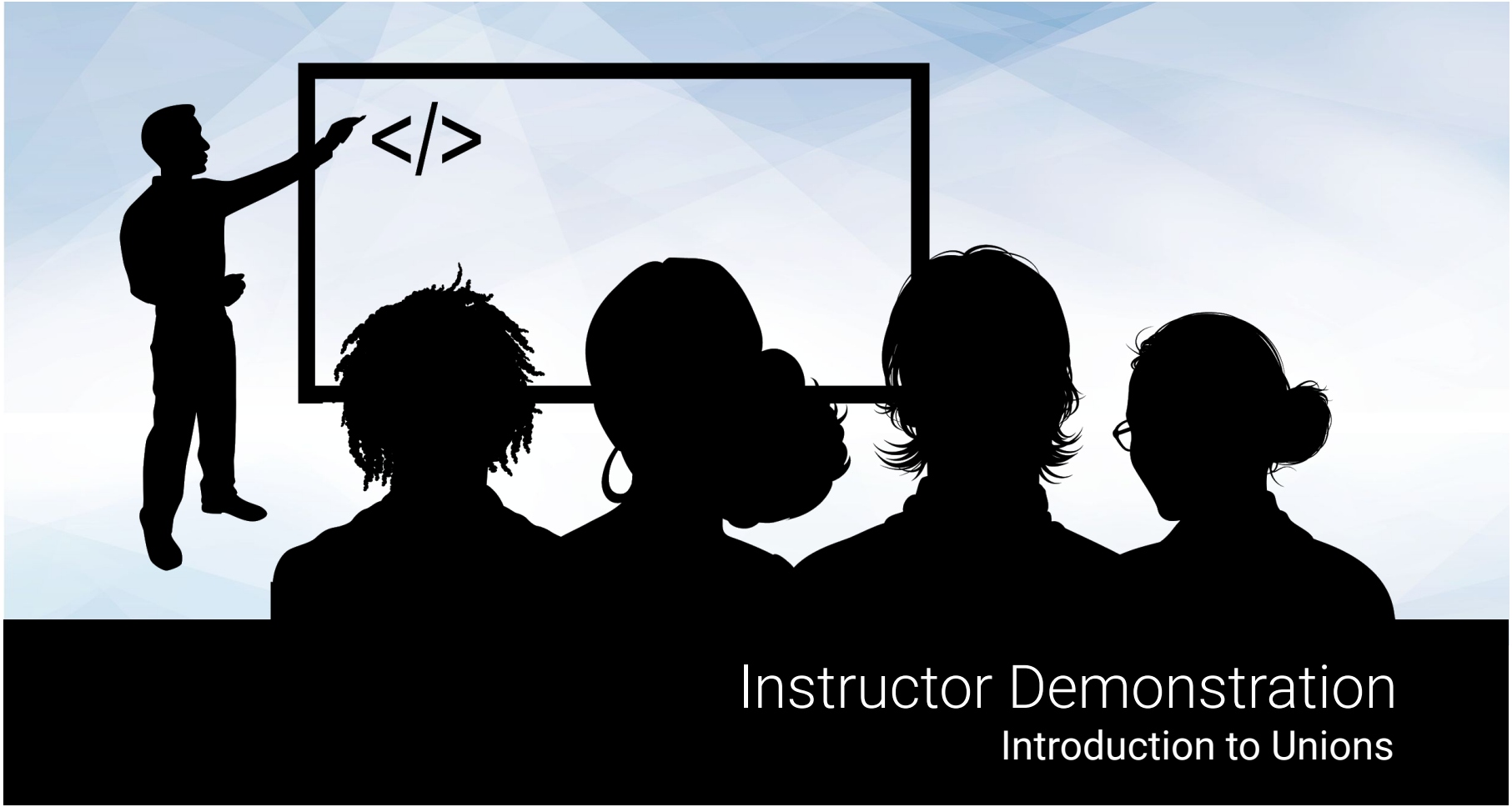
Activity: Designing an ERD, Part II

- **Hints**
  - Foreign keys are added to each table represented by the FK acronym, followed by the relationship, e.g., `OrderID INT FK >- Order.OrderID`.
  - You will need to add foreign keys to your tables in order to map the data relationships
  - Remember to document the relationships between entities using the correct symbols. Here are the allowed relationship types:

- Keep in mind the following:
  - Each member belongs to only one gym.
  - Trainers work for only one gym, but a gym has many trainers.
  - Each member must have a trainer, but each trainer may instruct multiple members.
  - Each member has one credit card on file.
- Once you have created tables in pgAdmin, you can check the table creation with the following syntax: `SELECT * FROM Members;`



Time’s Up! Let’s Review.

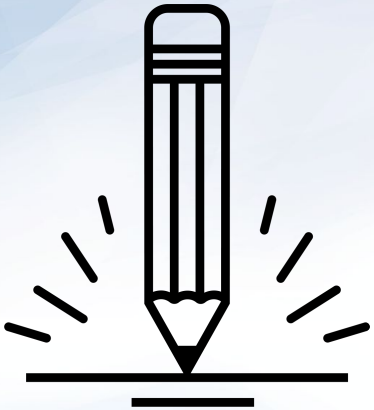


Instructor Demonstration

Introduction to Unions

Introduction to Unions

<Time to code>



Activity: Unions

In this activity, you will practice more with unions, by combining data from multiple tables without the use of joins.

Suggested Time:
15 Minutes

Activity: Designing an ERD, Part II

- Using `UNION`, write a PostgreSQL statement to query the number of rows in tables `city` and `country`.
- Use `UNION` to display from the tables `customer` and `customer_list` the ID of all customers who live in the city of London. Determine whether both tables contain the same customers by using `UNION ALL`.
- **Hint:**
  - For the second problem, consider using subqueries.

Time’s Up! Let’s Review.


The End