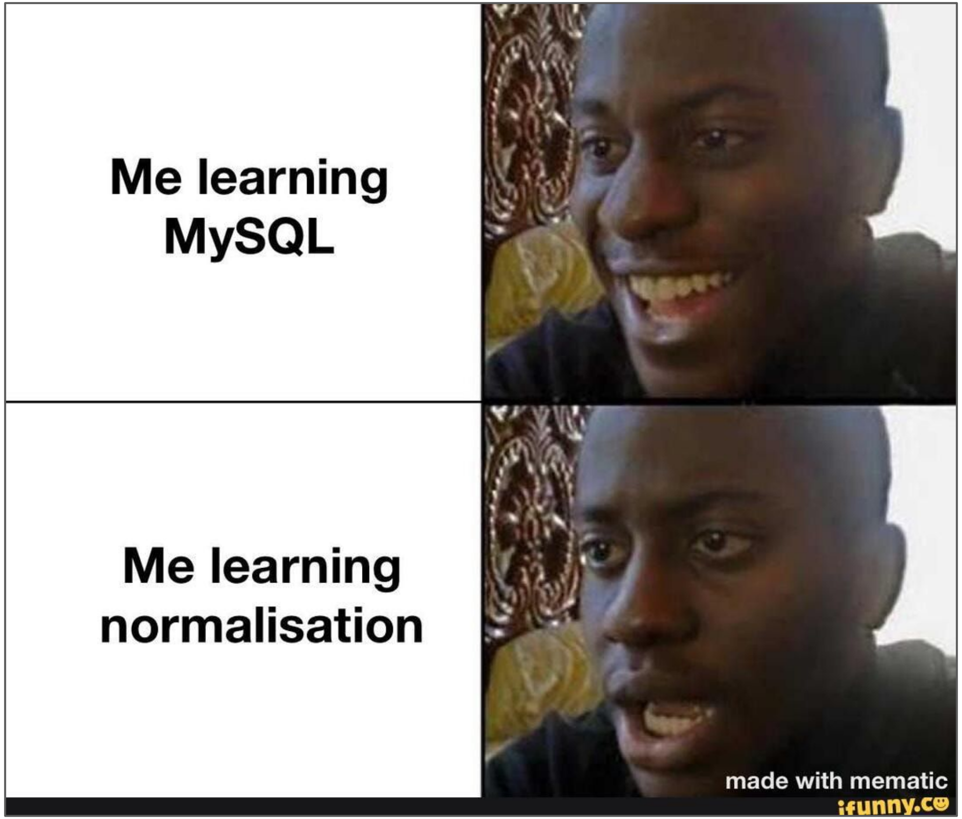


Image Source

Welcome to Week 13 Lecture 2!

MySQL/MySQL Workbench,
Database Design & Administration

Agenda

- Review: Relational Databases
- Review: Basic SQL Queries
- CodeAlong: Querying MySQL with Python
- Database Design/Administration
- Breakout Room Activity: Database Design

Assignments

- Core Assignments for this Week:
  - Queries: Sakila (Core)
  - Project 3 - Part 1 (Core)
  - Books (Core)

- Assignment Deadline - This Week:
  - Deadline is Friday at 9 AM PST.
  - If you have technical issues, email me (jirving@codingdojo.com) by 9 AM PST with:
    - Which assignment you’re having trouble with
    - A description of your issues/errors.
    - If its an error, include screenshots!

Quick Announcements

- Check the Helpful Links - Stack 4 sheet in our Stack 4 Schedule
  - New Notes Repository and Playlist added
- Code Review Sign-Ups
- Out of Office on Friday

Relational Databases

What is a Relational Database?

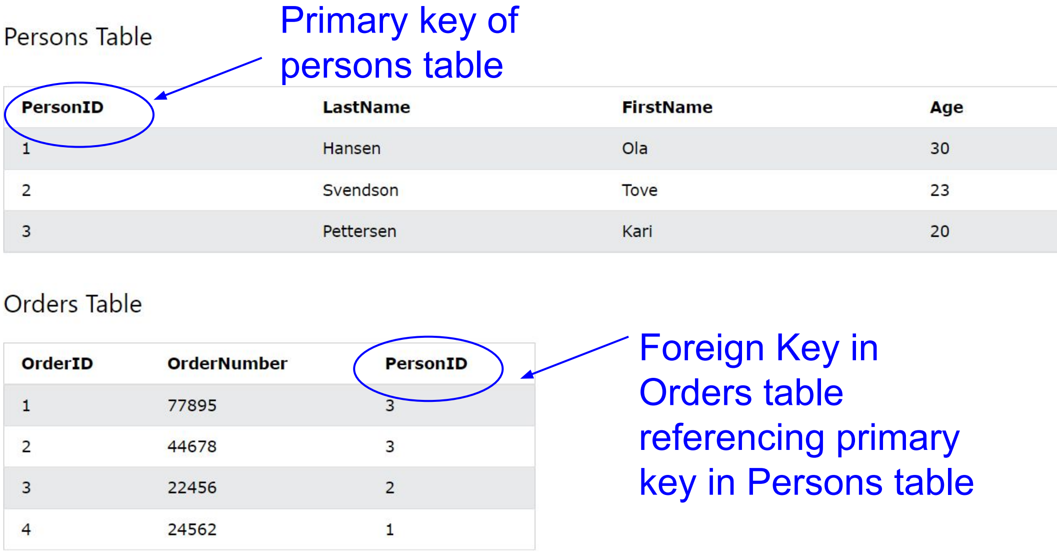
- A relational database is a data storage system that contains multiple tables that can be linked to each other (they are *related)*.
    - The tables are linked via “keys”
- Relational Databases are good for:
    - Highly structured data.
    - Reducing redundancy in data.

- SQL is the programming language used by most relational databases.

Keys link one table to another

- Primary key columns:
  - All unique values - no repeats (like an ID number).
  - Cannot contain null values.
  - Only one primary key per table.

- Foreign keys
  - References the primary key in another table.
  - Used to match related data across tables.



Persons Table

Primary key of persons table

PersonID	LastName	FirstName	Age
1	Hansen	Ola	30
2	Svendson	Tove	23
3	Pettersen	Kari	20

Orders Table

OrderID	OrderNumber	PersonID
1	77895	3
2	44678	3
3	22456	2
4	24562	1

Foreign Key in Orders table referencing primary key in Persons table

Notice how every time an order is placed, we don't need to repeat all the customer info. We can just link to it with the foreign key.

Image from W3Schools

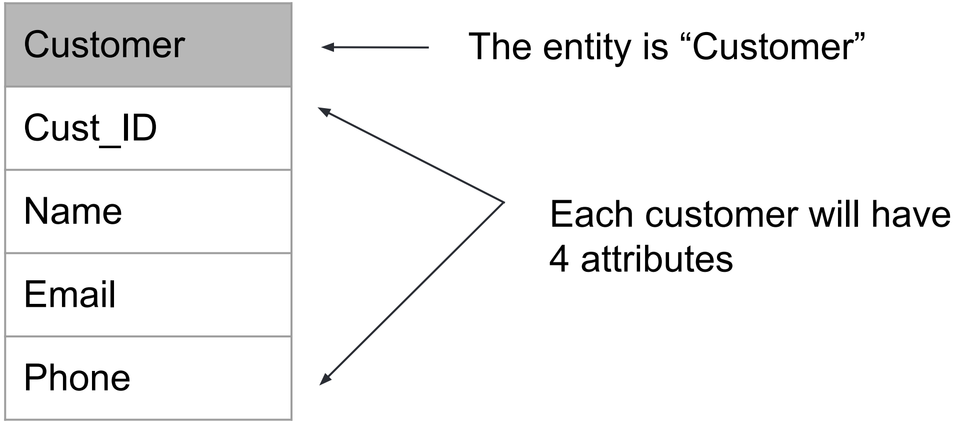
Entity Relationship Diagram (ERD)-provides an overview of all the tables and relationships

- An entity is essentially anything that can have information stored about it: It can be a person, thing, concept, or event.

- In a relational database, each “entity” has its own table.

- Each table contains the attributes associated with the entity

Entity
Attribute
Attribute
Attribute



Customer
Cust_ID
Name
Email
Phone

The entity is “Customer”

Each customer will have 4 attributes

Types of Table Relationships

“Cardinality” (in database design) is how many times can an entity exist in relation to another entity. In other words, it is how many possible matches can there be for any individual ID.

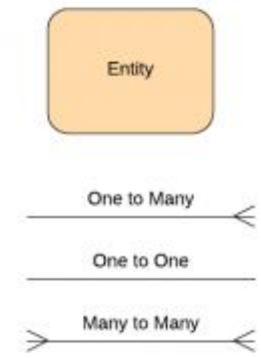
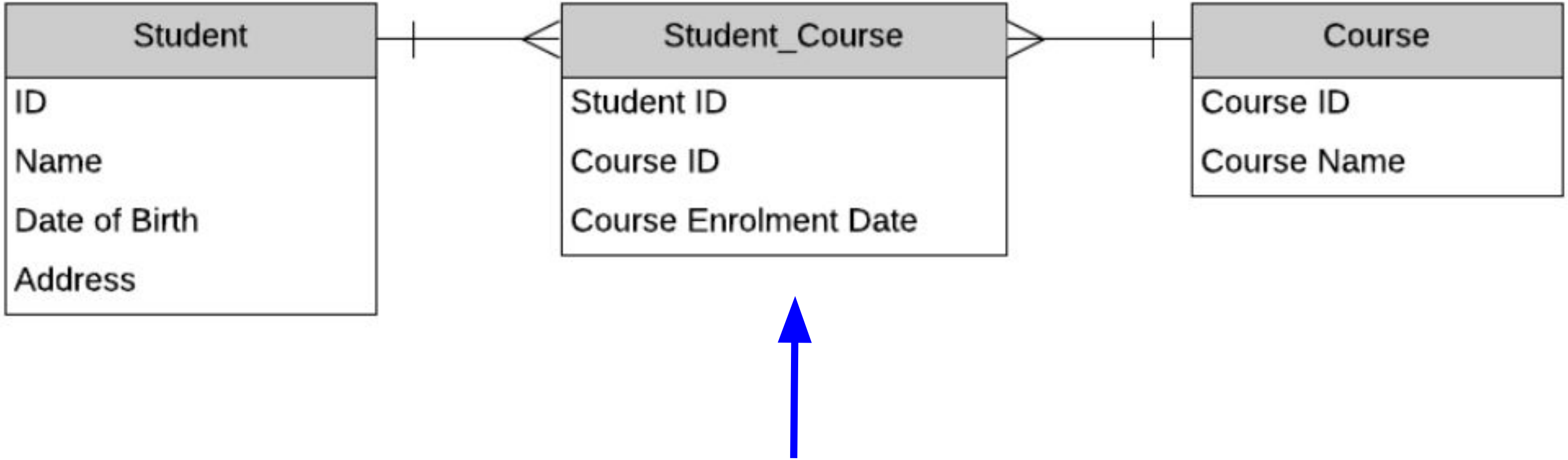


Image Source

- One to one (1:1) relationships
  - One record only associated with one record in another table
  - Example: Each Cust_ID in the Customers table can only be related to one entry in the Contact table
- One to many (1:M) relationships
  - One record can be associated with more than one record in another table
  - Example: Each Cust_ID can be associated with multiple orders
- Many to Many (M:M) relationships
  - One record from either table can be associated with many records from another table.
  - Example: A student can enroll in multiple courses, and a course can have multiple students

Check out this link at Database Star for examples of notation methods for representing cardinality and different types of ERDs.

Example ERD



Student

ID

Name

Date of Birth

Address

Student_Course

Student ID

Course ID

Course Enrolment Date

Course

Course ID

Course Name

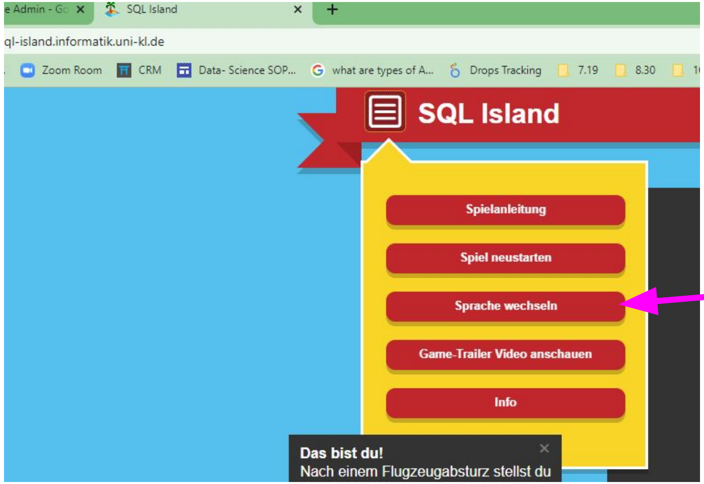
Joiner Table

Source is databasestar.com

Reviewing Basic SQL Queries

Resources/Practice

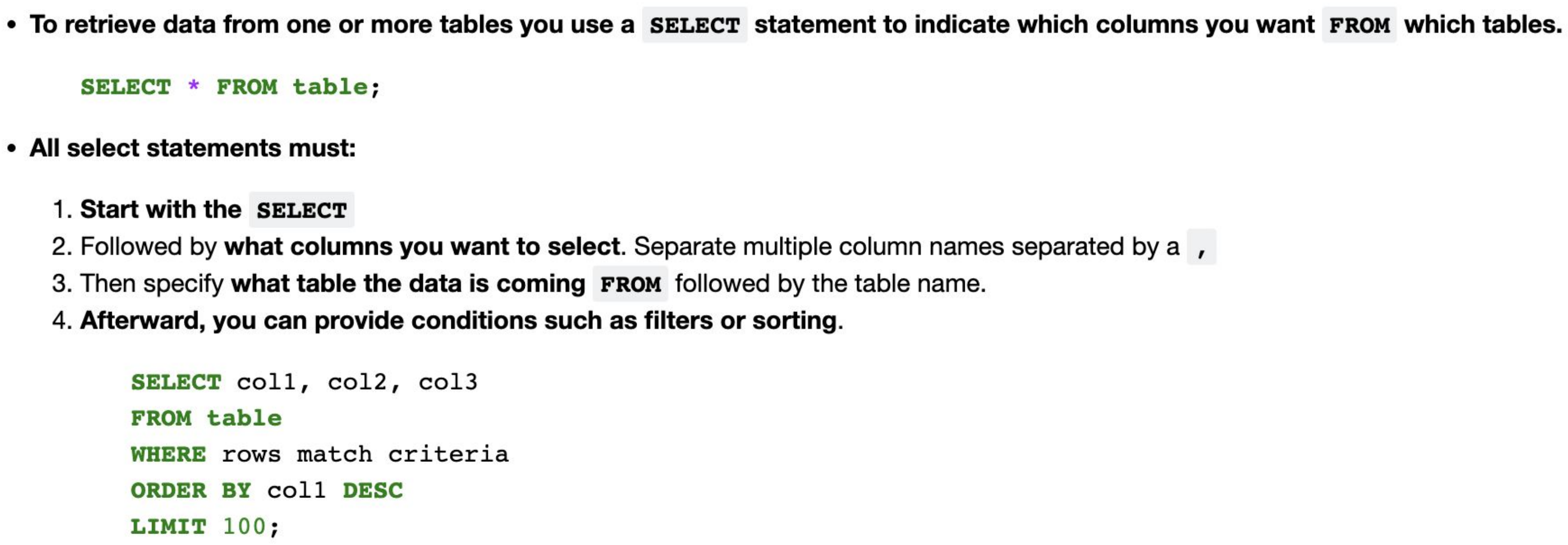
- Guided Practice on Basic SQL Queries: https://sqlzoo.net/wiki/SQL_Tutorial
- Good SQL Cheat Sheet: Google Drive Link
- Game to Practice SQL: SQL Island (Note: defaults in German)



Das bist du!
Nach einem Flugzeugabsturz stellst du

To change languages

Querying Databases - SELECTing data



- To retrieve data from one or more tables you use a `SELECT` statement to indicate which columns you want `FROM` which tables.

```
SELECT * FROM table;
```

- All select statements must:

1. **Start with the** `SELECT`
2. Followed by **what columns you want to select**. Separate multiple column names separated by a `,`
3. Then specify **what table the data is coming** `FROM` followed by the table name.
4. **Afterward, you can provide conditions such as filters or sorting**.

```
SELECT col1, col2, col3
FROM table
WHERE rows match criteria
ORDER BY col1 DESC
LIMIT 100;
```

SQL Gotcha’s

- WHERE VS HAVING:
  - Use WHERE to filter for a specific condition.
  - Use HAVING to filter for a specific condition after a GROUP BY

- Aggregate Functions (SUM/COUNT/MIN/MAX/AVG)
  - Require a GROUP BY

- Aliasing is used before its defined!

```
SELECT c.name
FROM customers AS c
```

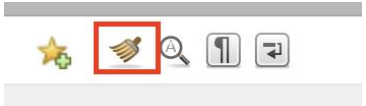
- AS is not required for table aliasing

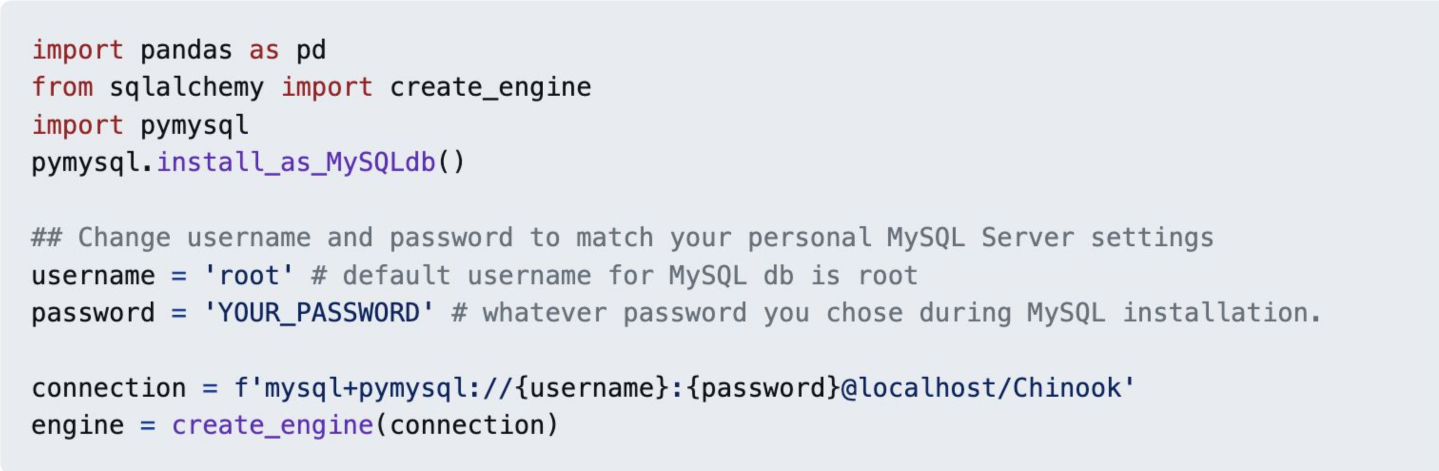
```
SELECT c.name
FROM customers c
```

CodeAlong:
Querying MySQL with Python

Connecting to MySQL with Python

- Use sqlalchemy to connect to the MySQL database, as demonstrated below.
- We can use MySQL Workbench to test-drive your queries!
  - Note: try the beautify button!





```
import pandas as pd
from sqlalchemy import create_engine
import pymysql
pymysql.install_as_MySQLdb()

## Change username and password to match your personal MySQL Server settings
username = 'root' # default username for MySQL db is root
password = 'YOUR_PASSWORD' # whatever password you chose during MySQL installation.

connection = f'mysql+pymysql://{username}:{password}@localhost/Chinook'
engine = create_engine(connection)
```

CodeAlong Details

- **For today’s activity, you are going to be practicing working with GitHub Desktop, Jupyter Notebook, and MySQL Workbench.**
  - Full instructions are in the README: https://github.com/coding-dojo-data-science/data-enrichment-wk13-l01-activity
  - Brief Summary:
    - Fork and clone the GitHub repository:
    - Open the repo with Jupyter and create a new notebook.
    - Install the Chinook database into your MySQL Server.
    - Use Reverse Engineering in MySQL Workbench to create an ERD for the Chinook database you just installed.
    - Use PyMySQL and Sqlalchemy to perform the correct queries to answer the listed queries.

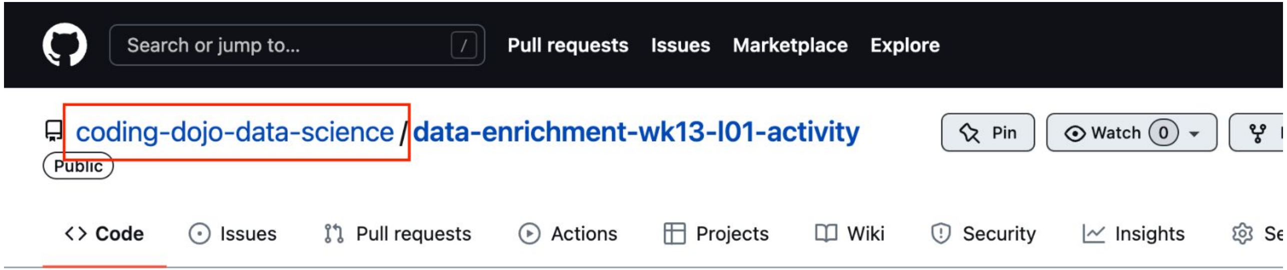
We will answer as many of the questions that we can as a group, while still leaving time for Database Administration.

Forking & Cloning Someone Else’s Repository

GitHub Repositories - Ownership

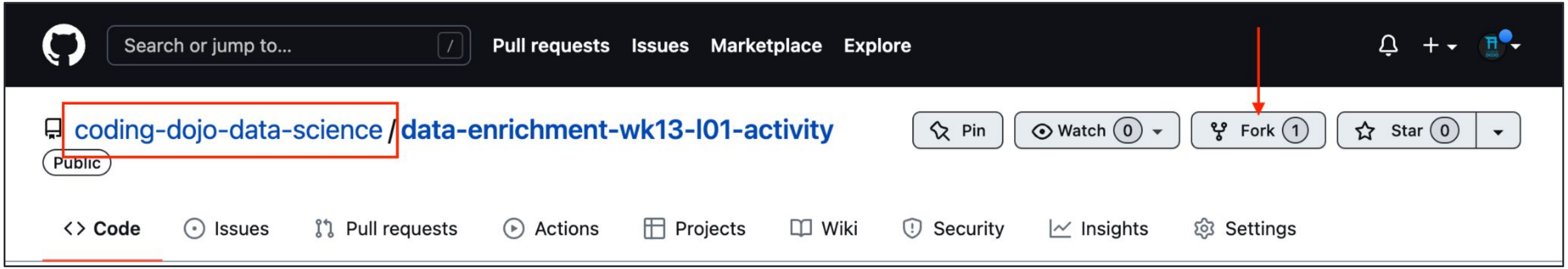
- **Repositories are owned by a specific GitHub user.** Only that user can change the contents of a repo. In the screenshot below, you can see that this repo is owned by "coding-dojo-data-science".

- For you to be able to save any changes made to a repository, YOU must be the owner.
  - To make a copy of someone else's repo that you own, **we "fork" a repo.**

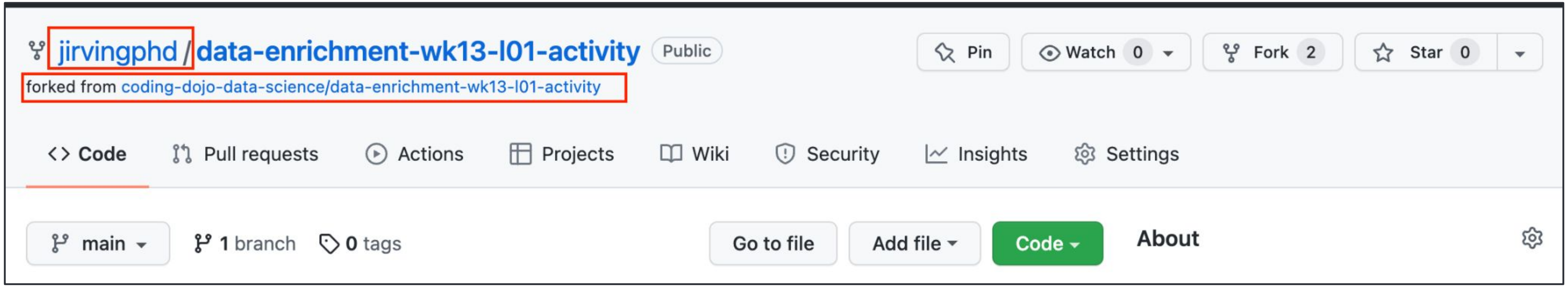


Forking a Repo

- To Fork a repo, click on the “Fork” button on the repo on GitHub.com

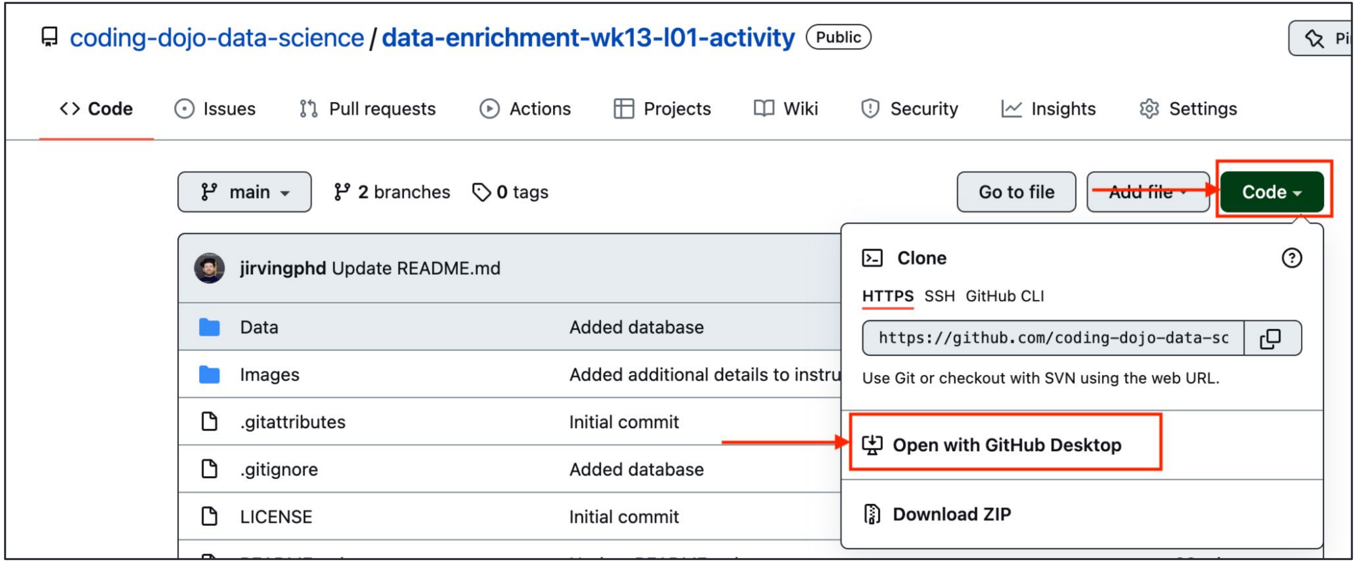


- It will then open a new copy of the repository, but attached to YOUR username. It will also indicate what repository it was forked from.



Opening a Repo Locally - 1

1. Click on green Code button and select Open in GitHub Desktop

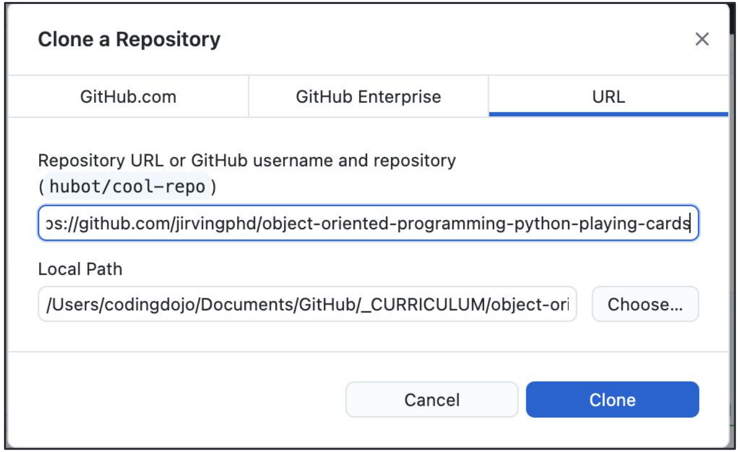


https://github.com/coding-dojo-data-sc

Use Git or checkout with SVN using the web URL.

Opening a Repo Locally - 2

2. Select a folder to clone the repo into and click “Clone”.



Clone a Repository

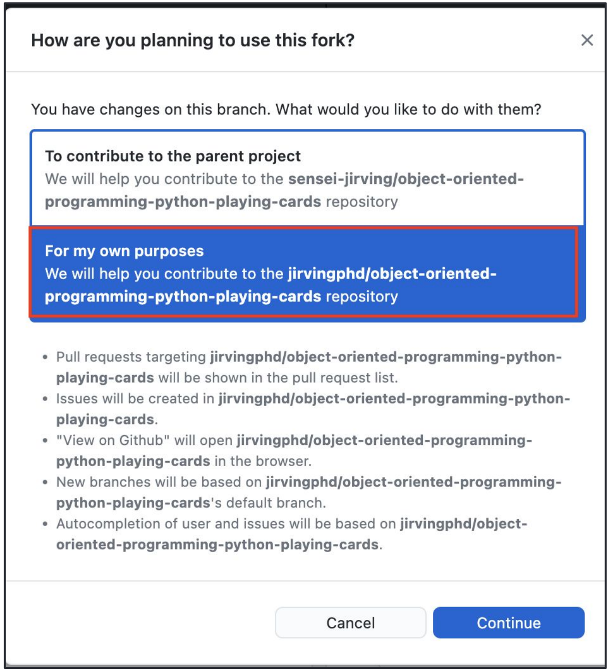
Repository URL or GitHub username and repository (`hubot/cool-repo`)

ɔs://github.com/jirvingphd/object-oriented-programming-python-playing-cards

Local Path

/Users/codingdojo/Documents/GitHub/_CURRICULUM/object-ori

2. When asked “How are you planning to use this fork”. Select “For my own purposes”



How are you planning to use this fork?

You have changes on this branch. What would you like to do with them?

To contribute to the parent project
We will help you contribute to the **sensei-jirving/object-oriented-programming-python-playing-cards** repository

For my own purposes
We will help you contribute to the **jirvingphd/object-oriented-programming-python-playing-cards** repository

- Pull requests targeting **jirvingphd/object-oriented-programming-python-playing-cards** will be shown in the pull request list.
- Issues will be created in **jirvingphd/object-oriented-programming-python-playing-cards**.
- "View on Github" will open **jirvingphd/object-oriented-programming-python-playing-cards** in the browser.
- New branches will be based on **jirvingphd/object-oriented-programming-python-playing-cards**'s default branch.
- Autocompletion of user and issues will be based on **jirvingphd/object-oriented-programming-python-playing-cards**.

Database Normalization

Database Normalization

- *“**Database normalization** is simply a convention for splitting large tables of data into smaller separate tables with the primary goal being to not repeat data.”*

There are three conventions/”Forms” for data normalization:

- 1st Normal Form(1NF):
  - Each column can only have 1 piece of information.
  - No lists of values.

- 2nd Normal Form(2NF):
  - Each column must have unique values for every row, with the exception of foreign key columns.

- 3rd Normal Form (3NF):
  - No non-key column is dependent on another non-key column.
  - Each column is describing something about the associated primary key.

Appointments Flat File

Patient Name	Patient Phone	Date of Appointment	Reason	Doctor Seen	Location	Location Address	Charge Tier	Charge
Sandy Summers	855-100-1224	4/21/2022	Splinter	Dr. Who	North Office	1000 Main St, Greenville, SC	Low	50
Angel Autumn	855-200-5476	4/21/2022	Headache Cough Runny Nose Splinter	Dr. When	North Office	500 Oak St, Greenville, SC	Medium	100
Wendy Winter	855-986-6548	4/20/2022	Appendix	Dr. Who	South Office	1000 North Main St, Greenville, SC	High	200

Looking Back at the Previous Slide’s Flat File

- Q1: Which column(s) violate the 1NF conventions?
- Q2: Which columns(s) violate the 2NF conventions?
- Q3: Which columns(s) violate the 3NF conventions?
- Thinking about how the information above, what separate tables might we create to make a relational database that doesn’t violate these assumptions?

Breakout Rooms - Group Activity

- Using MySQL Workbench, make an ERD for the Appointments Flat File
    - Consider how to best meet the conventions of normalization
    - Consider which type of relationships are needed
        - Note that a patient can have multiple reasons for a visit
        - Also note that a doctor can work in multiple locations
        - A low charge tier *always* costs $50, medium $100, and high $200
- With MySQL Workbench, it is often easier to “think it through” as you create the model!
- Once you have designed the model, save it!
- Forward engineer it to create (an empty) database!

SOLUTION WILL BE SHARED ON DISCORD TOMORROW!