



# SQLAlchemy ORM

Data Boot Camp



# Today's Goals

---

By the end of this class, you will:



Use SQLAlchemy ORM to model tables.



Perform CRUD with SQLAlchemy.



Reflect existing databases with SQLAlchemy.



Plot query results from SQLAlchemy ORM.



Run a t-test to validate differences in means.



# Instructor Demonstration

## SQLAlchemy Queries In Action

# Lets Run through a Review!





How can you query  
a database  
using SQLAlchemy?

# There are two ways to query a database using SQLAlchemy

---

## Using SQL...

```
data = engine.execute("SELECT * FROM BaseballPlayer")
```

## ...or Python!

```
players = session.query(BaseballPlayer)
for player in players:
    print(player.name_given)
```



What is a t-test,  
and what is it used for?

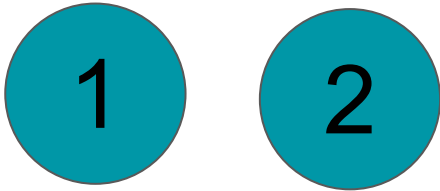
# A t-test is used to test the difference between means!

---

There are two types of (two-sample) t-tests

01

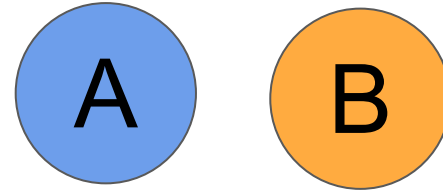
Paired



- Compares the means of the **same** group
- Example:
  - Mean blood pressure before and after medication

02

Unpaired



- Compares the means of **different** group
- Example:
  - Cost of restaurant dinners in Minnesota vs. Texas



# <Time to Code>





## Activity: Sharks Search

In this activity, you will create a Python script that can search through the SQL file of shark attacks provided.

**Suggested Time:**  
20 Minutes



# Sharks Search Instructions

---

- Within a Python script, create a Sharks class that will be able to read all of the columns in from the table you created
- Using SQLAlchemy, perform the following queries...
  - Print all locations of shark attacks
  - Find the number of provoked attacks
  - Find the number of attacks in the USA
  - Find the number of attacks in 2017
  - Find the number of attacks while surfing
  - Find the number of fatal attacks
  - Find the number of fatal attacks while surfing
  - Find the number of fatal attacks in Mozambique while spearfishing





**Time's Up!** Let's Review.



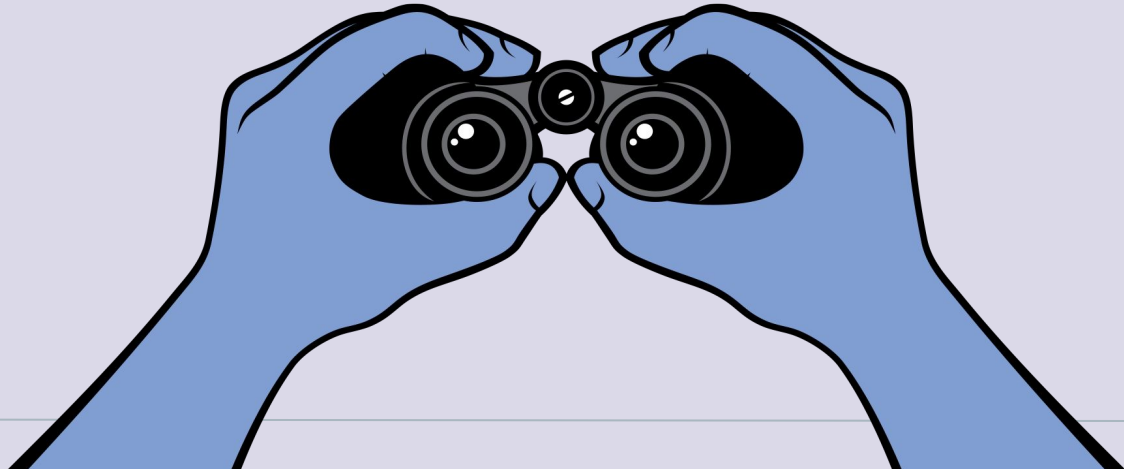
# Instructor Demonstration

## Updating and Deleting Rows

# We have only looked at one-half of CRUD!

---

C R U D  
create read update delete



# <Time to Code>





## Activity: What a Cruddy Database

In this activity, **you and a partner** will create a new SQLite database for a garbage collection company.

**Suggested Time:**  
20 Minutes





# What a Cruddy Database Instructions

---

- Within the unsolved Python file, create new SQLAlchemy class called Garbage that holds the values outlined in the Readme.md
- Create a connection and a session before adding a few items into the SQLite database crafted.
- Update the values within at least two of the rows added to the table.
- Delete the row with the lowest weight from the table.
- Print out all of the data within the database.





**Time's Up!** Let's Review.

A close-up, high-angle shot of a computer keyboard. The central focus is a large, white, rectangular key with rounded corners. On this key, there is a dark blue icon of a coffee cup with three wavy lines above it representing steam. Below the icon, the word "Break" is printed in a dark blue, serif font. The key is set against a light-colored, textured keyboard surface. Other keys are visible in the background, including one with a double quote symbol and another with a dash/slash symbol, but they are out of focus.

Break



# Instructor Demonstration

## Reflections

But how can  
we analyze  
databases that  
already exist?



**SQLAlchemy** provides tools  
for creating **ORM classes**  
for an existing database!





# Looking at our Reflection

---

```
# Python SQL toolkit and Object Relational Mapper  
import sqlalchemy  
from sqlalchemy.ext.automap import automap_base  
from sqlalchemy.orm import Session  
from sqlalchemy import create_engine
```

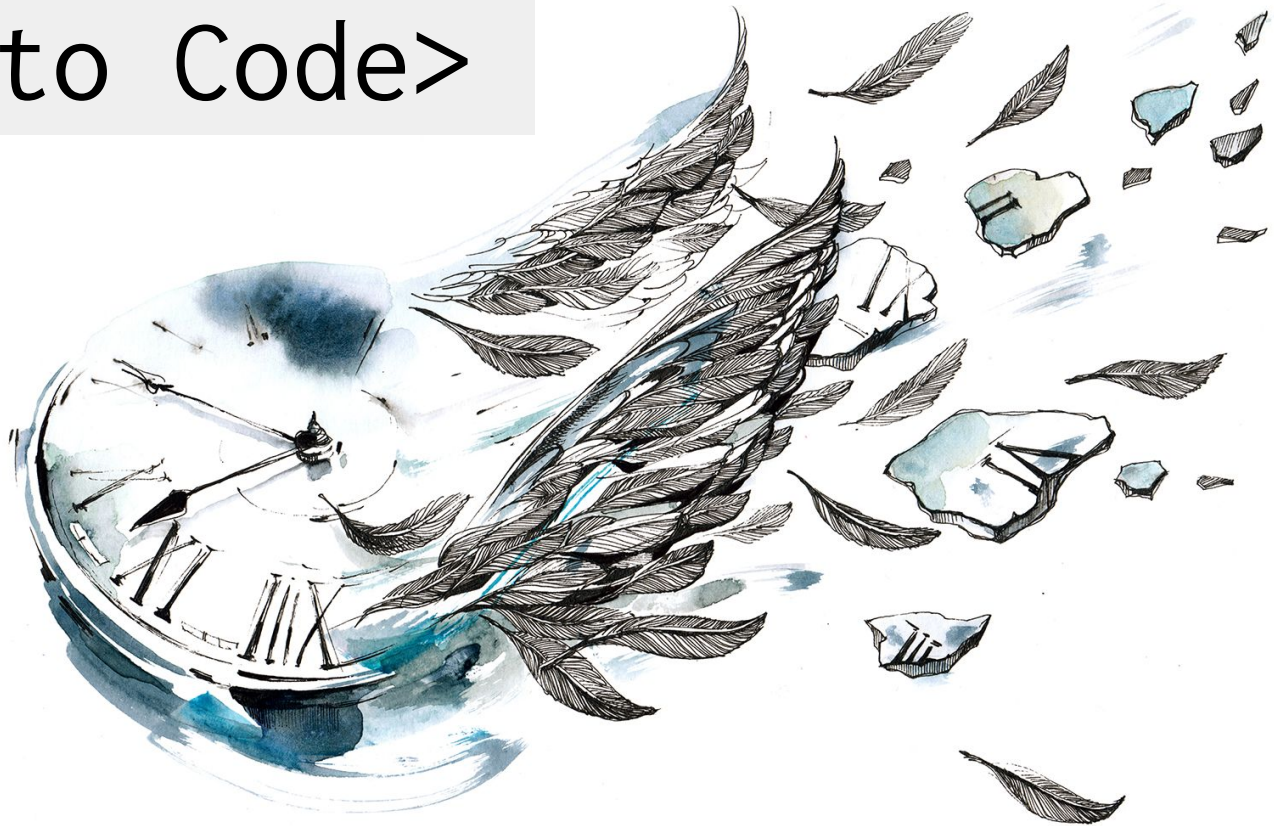
```
# Create engine using the `demographics.sqlite` database file  
engine = create_engine("sqlite:///../Resources/dow.sqlite")
```

```
# Declare a Base using `automap_base()`  
Base = automap_base()
```

```
# Use the Base class to reflect the database tables  
Base.prepare(engine, reflect=True)
```

```
# Print all of the classes mapped to the Base  
Base.classes.keys()
```

# <Time to Code>







## Activity: Reflecting on SQL

In this activity, you will practice your ability to reflect existing databases using SQLAlchemy and a SQLite table focused upon demographic data.

**Suggested Time:**  
15 Minutes



# Reflecting on SQL Instructions

---

- Create engine using the demographics.sqlite database file
- Declare a Base using `automap_base()` and use this new Base class to reflect the database's tables
- Assign the demographics table/class to a variable called `Demographics`
- Create a session and use this session to query the `Demographics` table and display the first five locations





**Time's Up!** Let's Review.



# Instructor Demonstration

## SQLAlchemy Exploration

# Reflecting on Reflections

---



- Reflecting using SQLAlchemy does not provide users with information on what is being stored
- The creators of SQLAlchemy understood this
  - They also created an inspector tool
- Inspector is used to look up tables, columns and datatypes.

```
import sqlalchemy
from sqlalchemy.ext.automap import automap_base
from sqlalchemy.orm import Session
from sqlalchemy import create_engine, inspect
```

```
# Create the connection engine
engine = create_engine("sqlite:///../Resources/database.sqlite")
```

```
# Create the inspector and connect it to the engine
inspector = inspect(engine)
```

```
# Collect the names of tables within the database
inspector.get_table_names()
```

# <Time to Code>





## Activity: Salary Exploration

In this activity, you will create an inspector and search through a SQLite database of salaries from San Francisco.

**Suggested Time:**  
15 Minutes



# Salary Exploration Instructions

---

- Using the attached SQLite file, use an inspector to collect the following information...
- The names of all of the tables within the database.
- The column names and data types for the Salaries table.







**Time's Up!** Let's Review.



## Activity: Emoji Plotting

In this activity, you will join forces to create a plot based upon the data stored within a SQLite database.

**Suggested Time:**  
15 Minutes



# Emoji Plotting Instructions

---

- Use the inspector to explore the database and print out the table names stored within it.
- Using the inspector, print out the column names and types for each of the tables contained within the SQLite file.
- Reflect the database into a SQLAlchemy class and start a session that can be used to query the database.
- Using Matplotlib, create a horizontal bar chart and plot the emoji score in descending order. Use emoji\_char as the y-axis labels and plot only the top 10 emojis ranked by score
- Create the same kind of chart using Pandas to plot the data instead of Matplotlib.





**Time's Up!** Let's Review.