Northwestern
INFORMATION TECHNOLOGY

Databases, SQL, and Django

Scott Coughlin, Computational Specialist

Some slides adapted from Professor Michael Coughlin and Matthew Graham

July 19th 2022

Databases, SQL, and Django

Topics Covered

1. Servers, Databases, Schema, Relations, OH MY!
2. Doing Things the Structured Query Language (SQL) Way
3. How to use `docker-compose`

Servers, Databases, Schema, Relations, OH MY!

What is the server?

It is the physical (or virtual) machine where the database (and the application managing the database lives).

- “The database runs at/lives at xxxx.ciera.northwestern.edu”

What is a database?

- A structured collection of data residing on a computer system (server) that can be easily accessed, managed and updated.
- Data is organized according to a database model.
- A Database Management System (DBMS) is a software package designed to store and manage databases.

Some Popular Database Management System

- Different DBMS:
  - Microsoft/Sybase
  - MySQL
  - Oracle
  - **PostgreSQL**
  - Redis, Hadoop

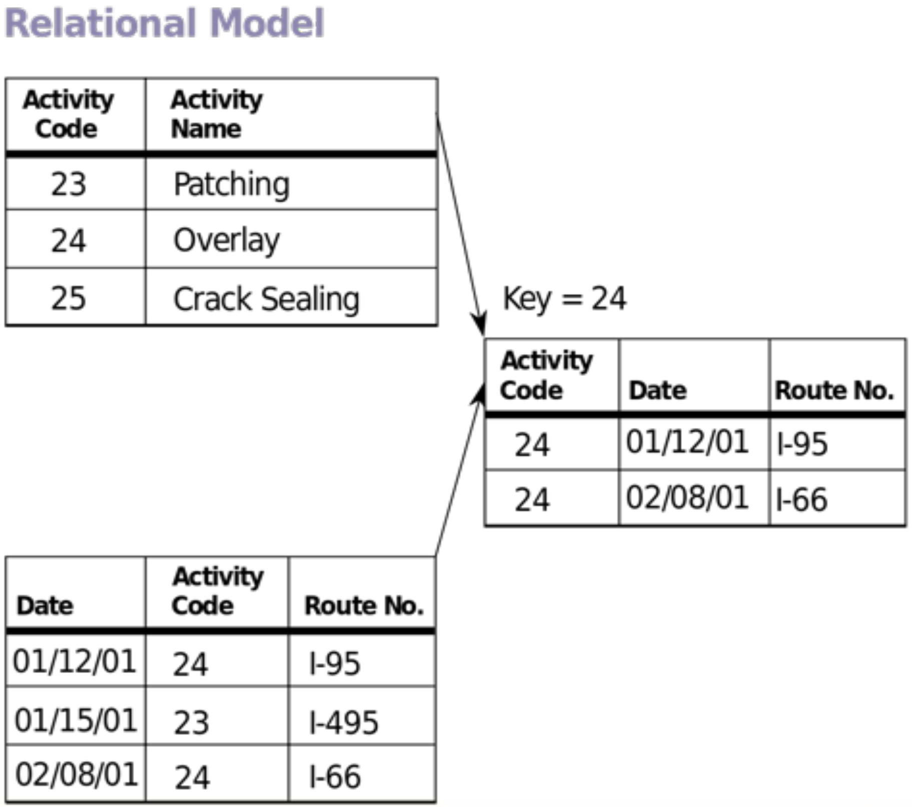
Why use a Database Management System?

- Provides concurrent access
- data scalability, expandability and flexibility
- Security through managing access to what databases, tables, and even types of queries a individual user can make.
- efficient memory management and indexing of the data
- Integrity constraints

How do we model our data? With tables (or relations)

- Data is organized as **relations (tables)**, **attributes (columns)** and **domains (type)**
- A **relation** is a table with columns (attributes) and rows (tuples)
- The **domain** is the set of values that the attributes are allowed to take
- Within the relation, each row is unique, the column order does not matter, and each row contains a single value for each of its columns

Relational Example



Relational Model

Activity Code	Activity Name
23	Patching
24	Overlay
25	Crack Sealing

Key = 24

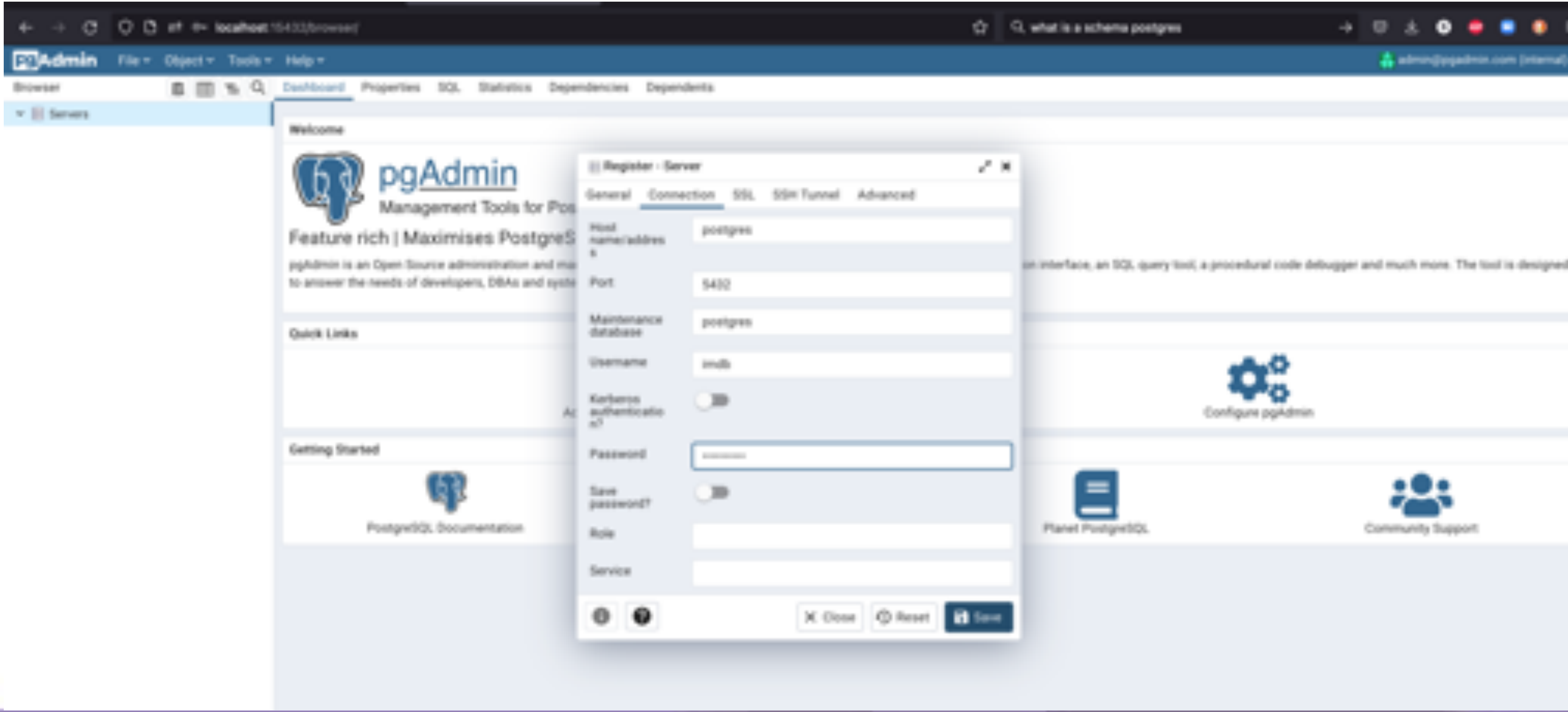
Activity Code	Date	Route No.
24	01/12/01	I-95
24	02/08/01	I-66

Date	Activity Code	Route No.
01/12/01	24	I-95
01/15/01	23	I-495
02/08/01	24	I-66

What is the “Schema”

- The **schema** is a named collection of tables.
- A schema can contain many additional pieces of information relevant to a collection of tables including views, indexes, sequences, data types, operators, and functions.

Say What Now?



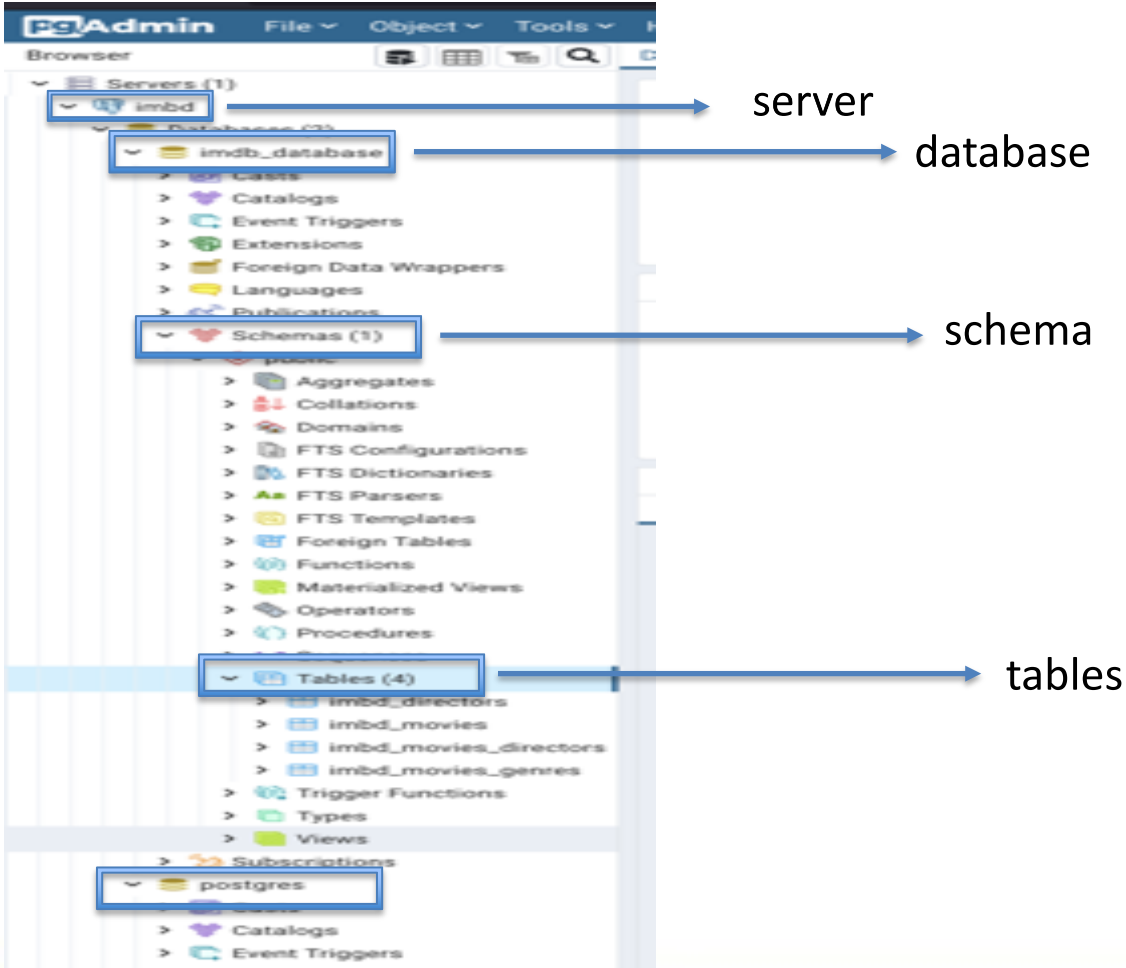
pgAdmin is an Open Source administration and ma... ...on interface, an SQL query tool, a procedural code debugger and much more. The tool is designed to answer the needs of developers, DBAs and syste...

Register - Server

General | Connection | SSL | SSH Tunnel | Advanced

Host name/address	postgres
Port	5432
Maintenance database	postgres
Username	imdb
Kerberos authentication?	○
Password	••••••••
Save password?	○
Role	
Service	

Say What Now?



server

database

schema

tables

Doing things the Structured Query Language (SQL) Way

Structured Query Language

- Different flavors:
    - **DBMS – Flavor of SQL**
    - Microsoft/Sybase - Transact-SQL
    - MySQL - MySQL
    - Oracle - PL/SQL
    - PostgreSQL - PL/pgSQL

SELECT

SELECT *column1, column2* FROM *table* WHERE *condition (*LIMIT *#ofrows)* ORDER BY *sort_expression [ASC | DESC];*

```
SELECT name, constellation FROM star WHERE dec > 0 ORDER
BY vmag;

SELECT * FROM star WHERE ra BETWEEN 0 AND 90;

SELECT DISTINCT constellation FROM star;

SELECT name FROM star LIMIT 5 ORDER BY vmag;
```

JOIN

- Inner join: combining related rows

```
SELECT * FROM imbd_movies as s INNER JOIN imbd_movies_genres as t ON
s.movie_id = t.movie_id;

SELECT * FROM star s, stellarTypes t WHERE s.stellarType = t.id;
```

- Outer join: each row does not need a matching row

```
SELECT * from star s LEFT OUTER JOIN stellarTypes t ON s.stellarType = t.id;

SELECT * from star s RIGHT OUTER JOIN stellarTypes t ON s.stellarType = t.id;

SELECT * from star s FULL OUTER JOIN stellarTypes t ON s.stellarType = t.id;
```

Aggregate Functions

COUNT, AVG, MIN, MAX, SUM

```
SELECT COUNT(*) FROM star;

SELECT AVG(vmag) FROM star;

SELECT stellarType, MIN(vmag), MAX(vmag) FROM star GROUP BY
stellarType;

SELECT stellarType, AVG(vmag), COUNT(id) FROM star GROUP BY
stellarType HAVING vmag > 14;
```

Create

- CREATE DATABASE *databaseName;*
- CREATE TABLE *tableName* (name1 type1, name2 type2, . . .);

```
CREATE TABLE star (name varchar(20), ra float, dec float, vmag float);
```

- Data types:
  - Boolean, bit, tinyint, smallint, int, bigint;
  - real/float, double, decimal;
  - char, varchar, text, binary, blob, longblob;
  - date, time, datetime, timestamp

```
CREATE TABLE star (name varchar(20) not null, ra float default 0, ...);
```

KEYS

A **primary key** is a unique identifier for a row and is automatically not null

```
CREATE TABLE star (name varchar(20), ra float, dec float,
vmag float, CONSTRAINT PRIMARY KEY (name));
```

A **foreign key** is a referential constraint between two tables identifying a column in one table that refers to a column in another table.

```
CREATE TABLE star (name varchar(20), ...,
stellarType varchar(8), CONSTRAINT stellarType_fk
FOREIGN KEY (stellarType) REFERENCES
stellarTypes(id));
```

Show and Describe

SHOW ...

```
SHOW INDEXES IN star;

SHOW WARNINGS;
```

DESCRIBE…

```
DESCRIBE star;
```

INSERT

INSERT INTO *table* VALUES(val1, val2, ...);

```
INSERT INTO star VALUES('Sirius', 101.287, -16.716,
-1.47);

INSERT INTO star(name, vmag) VALUES('Canopus’, -
0.72);

INSERT INTO star SELECT ...;
```

DELETE

DELETE FROM *table* WHERE *condition;*

DROP TABLE *table;*

```
DELETE FROM star WHERE name = 'Canopus';
DELETE FROM star WHERE name LIKE 'C_n%';
```

UPDATE

UPDATE *table* SET *column* = val1 WHERE condition;

```
UPDATE star SET vmag = vmag + 0.5;

UPDATE star SET vmag = -1.47 WHERE name LIKE 'Sirius';

UPDATE star INNER JOIN temp on star.id = temp.id SET star.vmag =
temp.mag;
```

ALTER

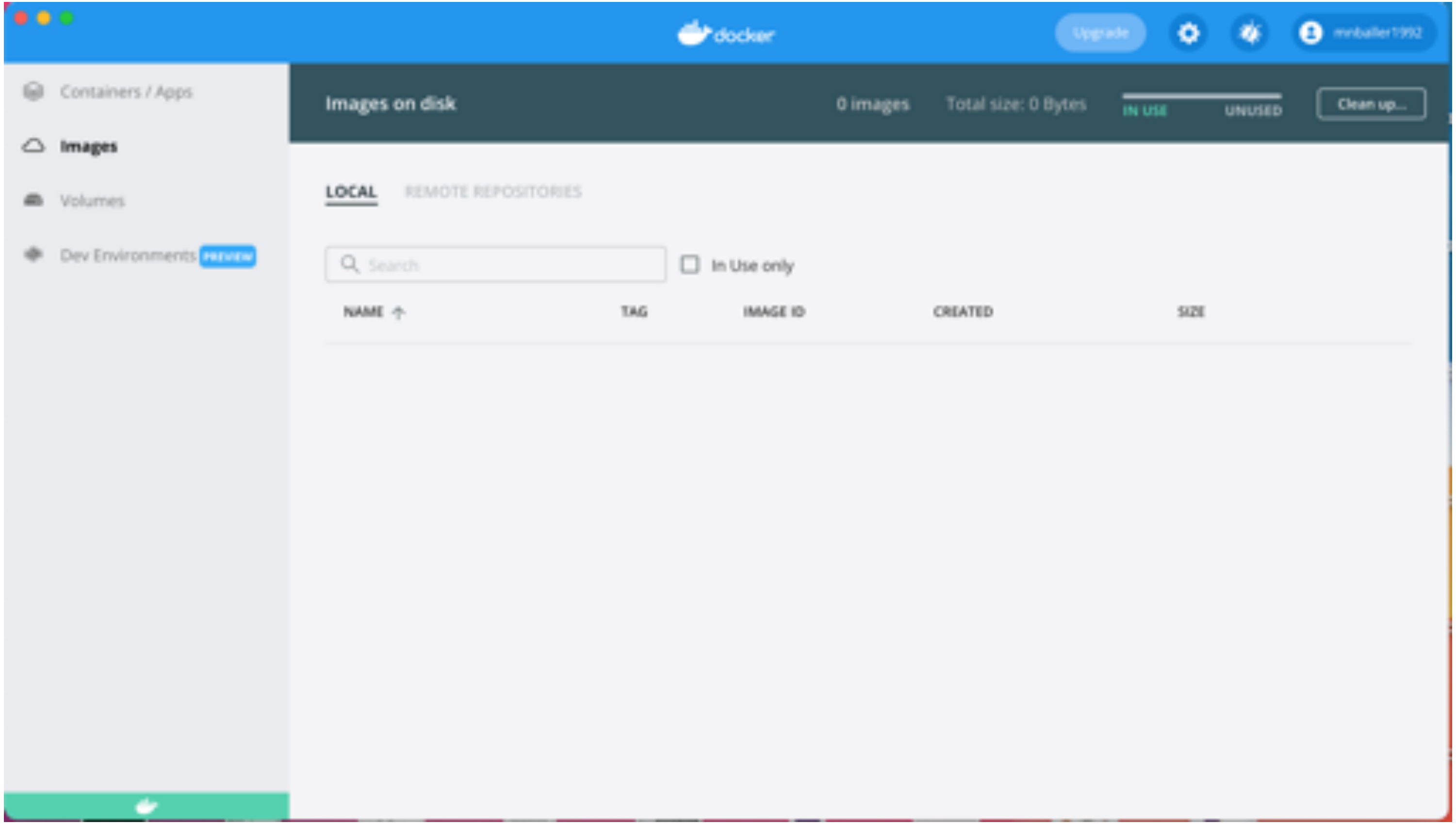
ALTER TABLE *table …;*

```
ALTER TABLE star ADD COLUMN bmag double AFTER vmag;

ALTER TABLE star DROP COLUMN bmag;
```

How to use `docker-compose`

Launch DockerHub



Images on disk

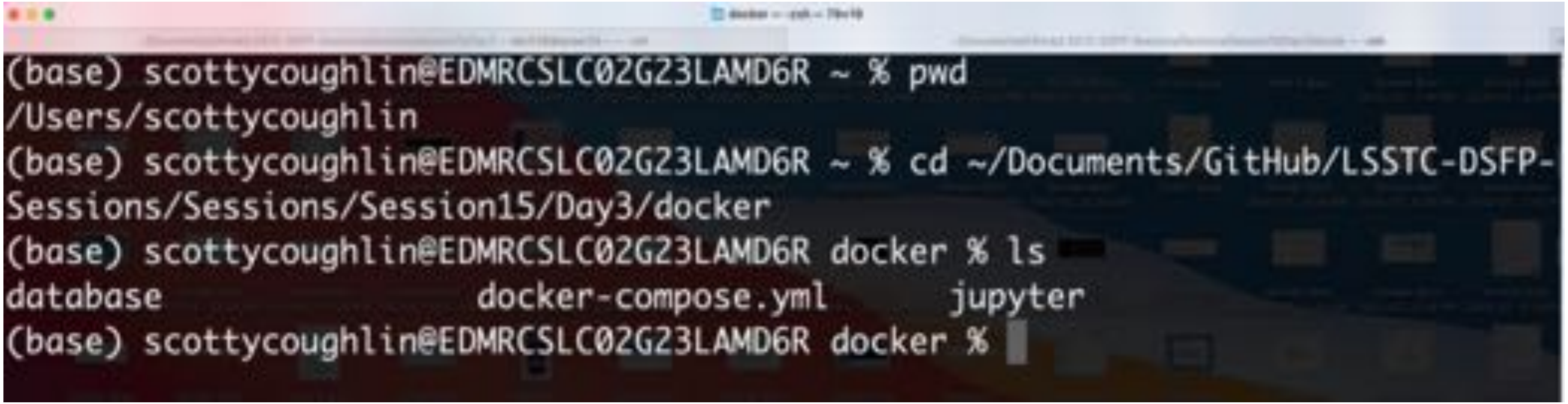
0 images

Total size: 0 Bytes

□ In Use only

NAME ↑	TAG	IMAGE ID	CREATED	SIZE

Open Terminal or PowerShell



```
(base) scottycoughlin@EDMRCSLC02G23LAMD6R ~ % pwd
/Users/scottycoughlin
(base) scottycoughlin@EDMRCSLC02G23LAMD6R ~ % cd ~/Documents/GitHub/LSSTC-DSFP-Sessions/Sessions/Session15/Day3/docker
(base) scottycoughlin@EDMRCSLC02G23LAMD6R docker % ls
database                docker-compose.yml      jupyter
(base) scottycoughlin@EDMRCSLC02G23LAMD6R docker % 
```

docker-compose

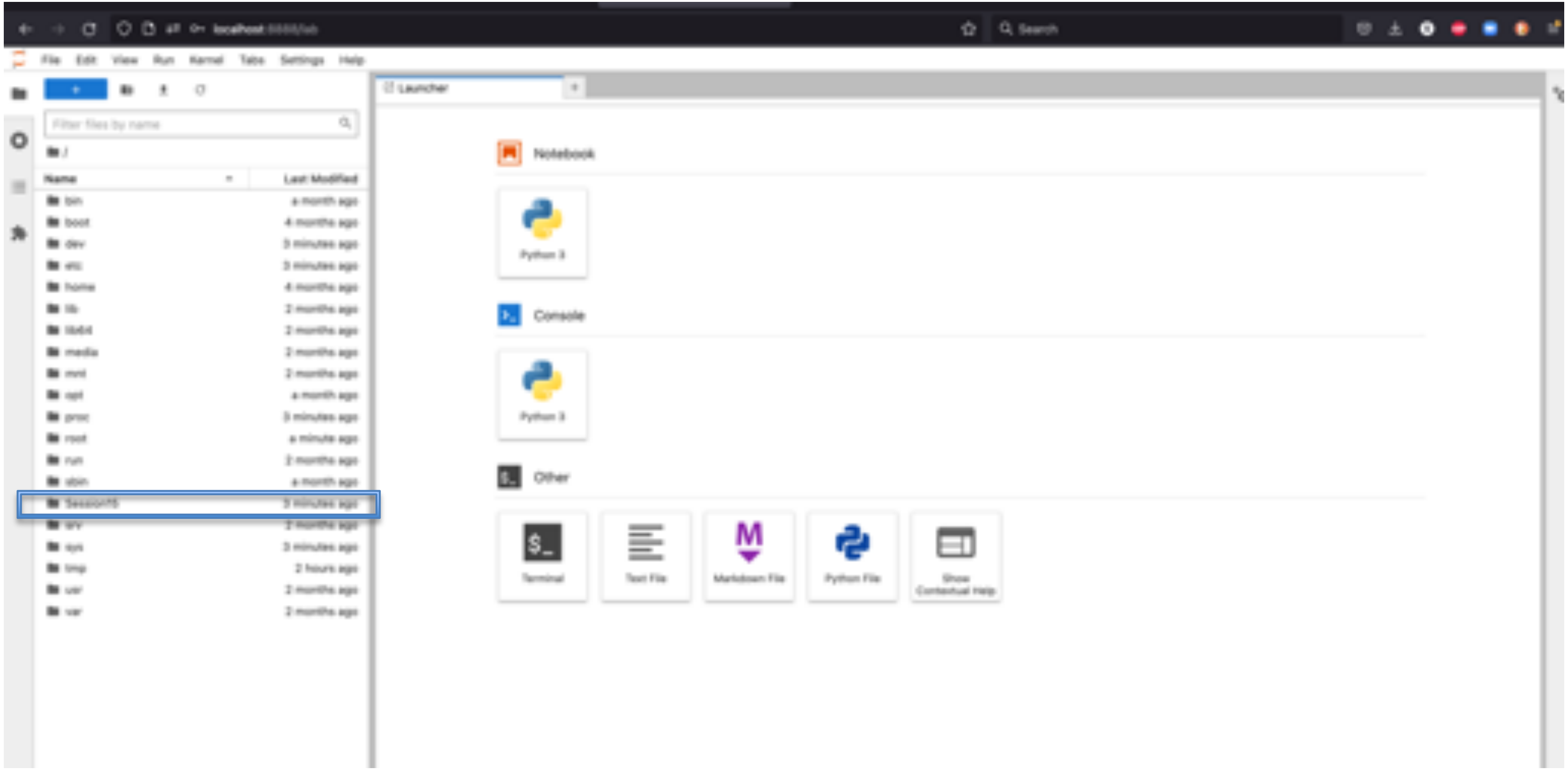
```
# Mac or Linux
$ cd LSSTC-DSFP-Sessions/Sessions/Session15/Day3/docker
# Power Shell
$ cd LSSTC-DSFP-Sessions\Sessions\Session15\Day3\docker
$ docker-compose up
```

- Three things are happening once you run this command!
  - A container with PostgresSQL installed is being downloaded, a database is being created and a table is being made for each “sheet” of IMBD data.
  - A useful web application, pgAdmin, is also being downloaded so that you can interact with PostGres via your browser.
  - A container with JupyterLab and Django is being made.

JupyterLab

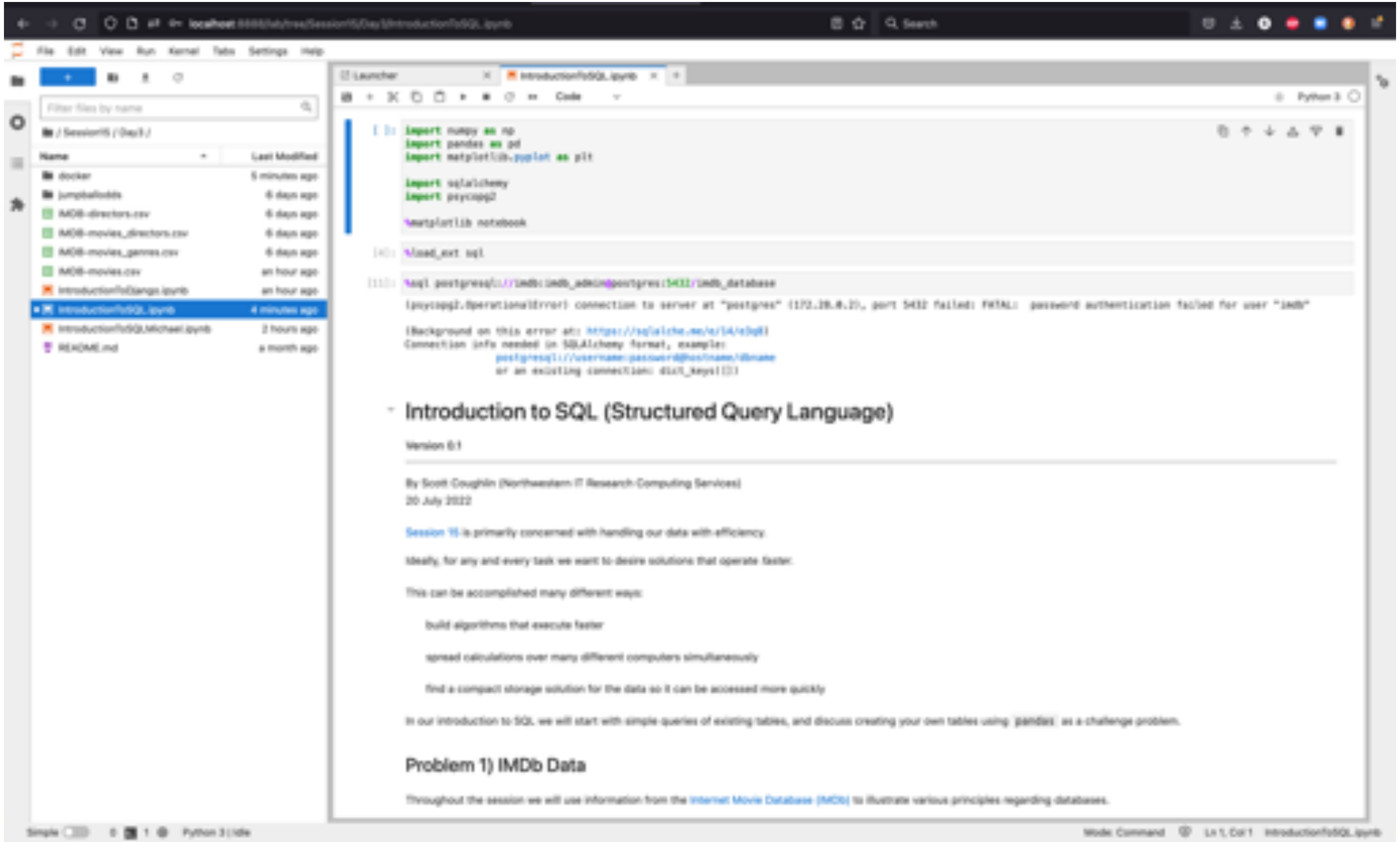
- Wait about 3 minutes and then…
- In your browser go to localhost:8888
- Password: Session15

JupyterLab



Name	Last Modified
bin	a month ago
boot	4 months ago
dev	3 minutes ago
etc	3 minutes ago
home	4 months ago
lib	2 months ago
lib64	2 months ago
media	2 months ago
mnt	2 months ago
opt	a month ago
proc	3 minutes ago
root	a minute ago
run	2 months ago
sbin	a month ago
Sessions	3 minutes ago
srv	2 months ago
sys	3 minutes ago
tmp	2 hours ago
usr	2 months ago
var	2 months ago

JupyterLab


localhost:8888/lab/tree/Session15/Day3/IntroductionToSQL.ipynb
Search
File Edit View Run Kernel Tabs Settings Help
Filter files by name
/ Session15 / Day3 /
Launcher
IntroductionToSQL.ipynb
Code
Python 3
Simple 0 1 Python 3 | Idle
Mode: Command Ln 1, Col 1 IntroductionToSQL.ipynb

Name	Last Modified
docker	5 minutes ago
jumpballodds	6 days ago
IMDB-directors.csv	6 days ago
IMDB-movies_directors.csv	6 days ago
IMDB-movies_genres.csv	6 days ago
IMDB-movies.csv	an hour ago
IntroductionToDjango.ipynb	an hour ago
IntroductionToSQL.ipynb	4 minutes ago
IntroductionToSQLMichael.ipynb	2 hours ago
README.md	a month ago

```
[ ]: import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt

     import sqlalchemy
     import psycopg2

     %matplotlib notebook
```

```
[4]: %load_ext sql
```

```
[11]: %sql postgresql://imdb:imdb_admin@postgres:5432/imdb_database
```

```
(psycopg2.OperationalError) connection to server at "postgres" (172.20.0.2), port 5432 failed: FATAL:  password authentication failed for user "imdb"

(Background on this error at: https://sqlalche.me/e/14/e3q8)
Connection info needed in SQLAlchemy format, example:
               postgresql://username:password@hostname/dbname
               or an existing connection: dict_keys([])
```

Introduction to SQL (Structured Query Language)

Version 0.1

By Scott Coughlin (Northwestern IT Research Computing Services)
20 July 2022

Session 15 is primarily concerned with handling our data with efficiency.

Ideally, for any and every task we want to desire solutions that operate faster.

This can be accomplished many different ways:

build algorithms that execute faster

spread calculations over many different computers simultaneously

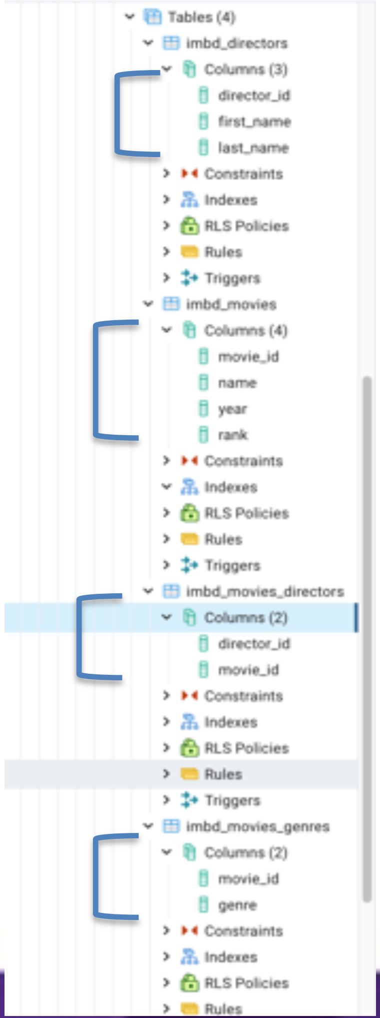
find a compact storage solution for the data so it can be accessed more quickly

In our introduction to SQL we will start with simple queries of existing tables, and discuss creating your own tables using `pandas` as a challenge problem.

Problem 1) IMDb Data

Throughout the session we will use information from the Internet Movie Database (IMDb) to illustrate various principles regarding databases.

JupyterLab



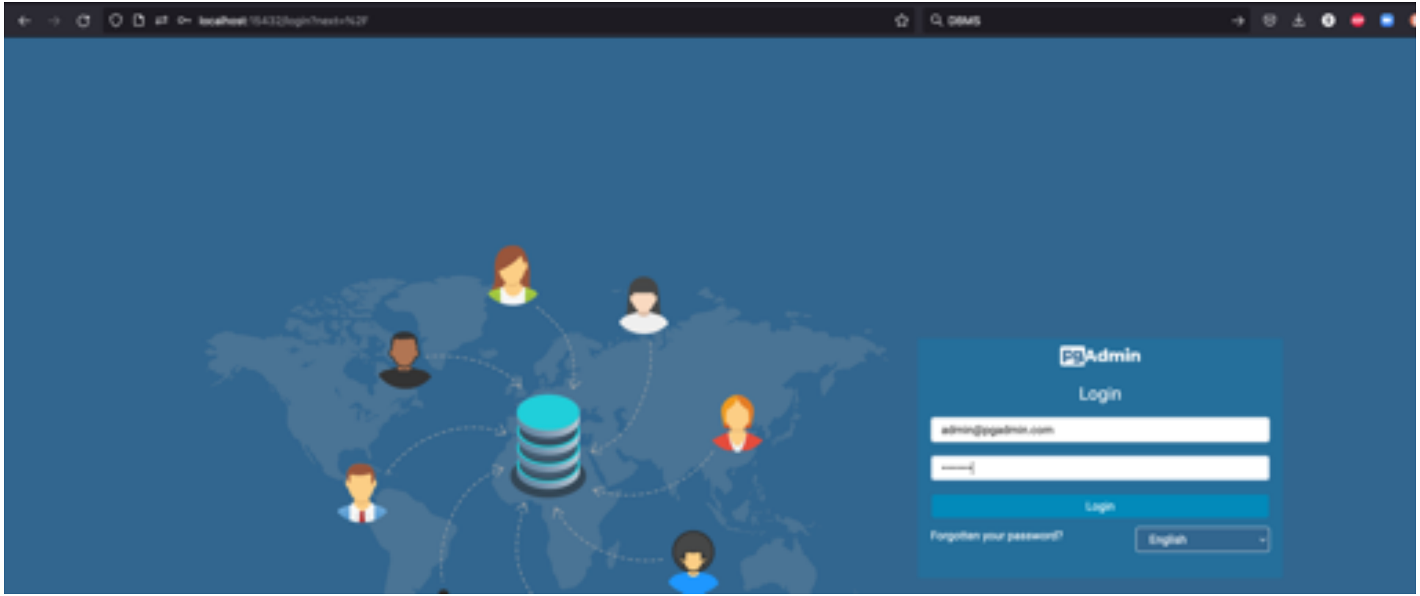
- Tables (4)
  - imbd_directors
    - Columns (3)
      - director_id
      - first_name
      - last_name
    - Constraints
    - Indexes
    - RLS Policies
    - Rules
    - Triggers
  - imbd_movies
    - Columns (4)
      - movie_id
      - name
      - year
      - rank
    - Constraints
    - Indexes
    - RLS Policies
    - Rules
    - Triggers
  - imbd_movies_directors
    - Columns (2)
      - director_id
      - movie_id
    - Constraints
    - Indexes
    - RLS Policies
    - Rules
    - Triggers
  - imbd_movies_genres
    - Columns (2)
      - movie_id
      - genre
    - Constraints
    - Indexes
    - RLS Policies
    - Rules

PGADMIN

In your browser go to `localhost:15432`

Username: **admin@pgadmin.com**

Password: **password**



localhost:15432/login?next=%2F

admin@pgadmin.com