Fairfield University
DOLAN
School of Business

Databases for Analytics

Course Onboarding

Syllabus, Software, Tutorials, etc.

Topics

- Introductions
- Why this course?
- Developers vs Analysts
- Course Syllabus, etc.
- Software Tools

Welcome

- Dr. Christopher L. Huntley
  - PhD in Systems Engineering (UVa, 1995)
  - At Fairfield U since 1997, before that (mostly) in industry
  - Mastered over a dozen programming languages so far
- Questions for you:
  - Who are you? (name, nickname, and hometown)
  - Background? (degrees and professional experience)
  - Something ***distinctive*** about yourself that we can't tell by looking at you?

The Big Picture

Databases and Business Analytics

Why Learn SQL? Aren't Excel, SAS, R, Python, etc. enough?

Analytical tools like Excel and Python have just about everything we need to analyze datasets (i.e, ***files***) from a variety of sources.

However, **sometimes data is found in *databases* instead of *files*.** This is especially true of live **transaction data** like that found in just about any corporate information system.

For that, we use SQL.

Transaction Data vs Analytical Data

	Transaction Processing	Analytical Processing
Example	Bank Accounts	Quarterly Financials
Age	Online/Live	Historical
Focus	Data Integrity & Controls	Informed Decision Making
Access	Multiple concurrent users Read and Write	Single user Read-only
Lang/Tech	SQL Database, Java, C#	Python, R, SAS, Excel

But can't we just use APIs?

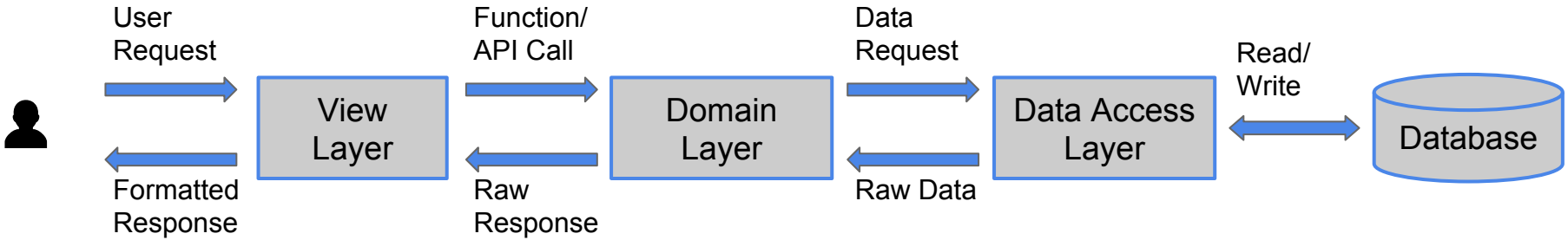
Many big corporate systems have **Application Programmer Interfaces (APIs)** that can be used to access data in real time.

- The programmers provide a function for every conceivable task one might want to ask the system to perform.
- Many of these functions are for data Creation, Retrieval, Updating, or Deletion (CRUD).

However, 'every conceivable task' does not mean full access to the data. You'll need SQL for that.

Three-Tiered Architecture

Virtually all modern information systems are organized into layers between the users and data.



User interface might be Jupyter Notebook

Processing might be in Java, PHP, or **Python**

Data requests are typically in **SQL**

Every write operation is a transaction

Goal: Unfettered Access to Data

So, to **ensure that you always get the most current and complete view** of transactional data (not just the functions in the APIs), you will need to **know and use SQL**.

Fortunately, SQL works great with just about any analytics platform: Python, R, Excel, Tableau, etc.

Course Expectations

What does success look like?

Knowing Our Limits

Knowing SQL is not the same thing as being a Database Engineer.

We only need to know enough SQL to …

- Get the data we need out of the system
- Manage (add/update/delete) the data in the system
- Perhaps suggest design changes to the system that would improve/simplify our analytical results

Course Objectives

- **Develop new skills**
  - Structured Query Language
  - Basic DB administration
- **Learn fundamentals of relational database systems**
  - Entity-Relationship Modeling
  - Relational model and table normalization
- **Apply knowledge and skills to business analytics**
  - Data Warehouse design project
  - (optional) Python integration with SqlAlchemy

Course Plans and Policies

Assignments, Grading, etc.

Coursework

- Tutorials (ungraded but required)
  - Cover specific theory and practice needed for the graded assignments. ***Progress is tracked online***.
- Quizzes (50% of course grade)
  - 5 Quizzes, with lowest grade dropped from Quiz Avg
- Team Project (40% of grade)
  - 2-3 students per team
  - Assigned in the fourth week of the course
- Professionalism (10% of grade)
  - Participation and timely completion of assigned work

Grading System: Curve *Everything*

Every graded assignment will be ***scored*** and then ***normalized*** using the following formula:

$$QP = 3.5 + \tfrac{1}{2}\,(x-\mu)/\sigma$$

The average QP is 3.5, which is an A-

where

- x is the student's raw score for the assignment
- μ and σ are the class average and standard deviation for the assignment

Letter grades are then 3.67+ → A, 3.34-3.66 → A-, …

Academic Honesty

- Cheating will be dealt with swiftly in accordance with Fairfield University policy
  - Unless given ***explicit permission*** to collaborate, do not share your work with others
  - *Avoid even the appearance of cheating!*
- Each graded assignment will be accompanied by the following (signed) pledge:
  - *On my honor as a Fairfield University student, I have neither given nor received any unauthorized aid on this assignment/quiz/project.*

Class Docs / Website

All lectures, programming assignments, etc. are available here:

https://christopherhuntley.github.io/ba510-docs

The class syllabus is linked from the home page:

https://christopherhuntley.github.io/ba510-docs/Syllabus.html

Setup

Accounts & Software Installation

If you took BA505 then please help the newbies.

Sign Up for DataCamp

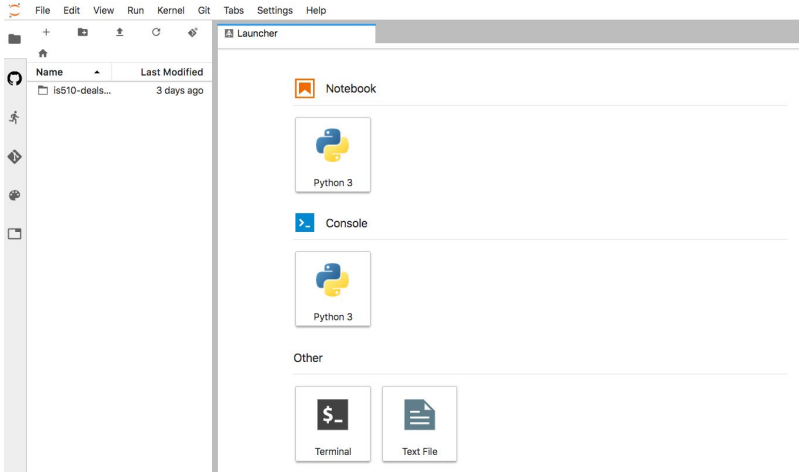
- Data Camp is an online school for data analytics in Python, R, and SQL. We have a “class group” for BA510 where your progress can be tracked.
- Invitation emails will be sent to your `fairfield.edu` address.
- Follow the instructions to confirm your enrollment on the class roster.

Claim Your Jupyter Account

Go to https://ba-lab.fairfield.edu

Log in as directed by Yue Pu.

After logging in you should see something like this:



GitHub / GitHub Classroom

All class documents, assignments, and projects will be managed online using GitHub.

- Syllabus, lectures, etc. are in the ba510-docs repo:
  - https://github.com/christopherhuntley/ba510-docs
- GitHub Classroom will be used to post and grade programming assignments
  - Invitations for each assignment will be sent by email
- We will more about GitHub as we go along, starting with a quick demo in class tonight

Sign Up for GitHub

1. Go to GitHub.com
2. Sign up for a new account using your Fairfield University email address.
3. Send an email from your student email to chuntley@fairfield.edu with your GitHub account username. The email subject is “GitHub account”.

Skip steps 1 and 2 if you already have a GitHub account **linked to your fairfield.edu email address**.

GitHub Classroom Roster

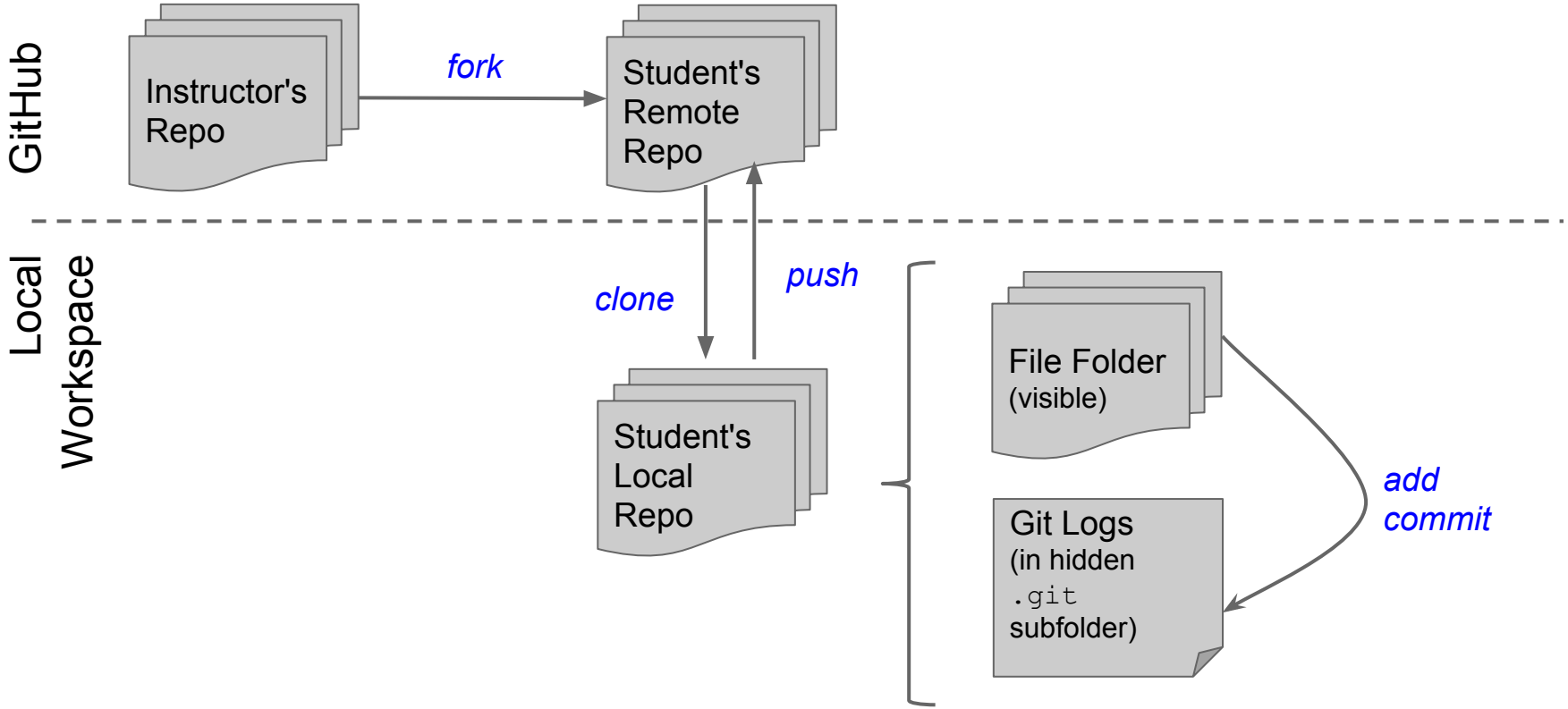
Take a break while the newbies are added to the class roster. (This has to be done manually. Ugh.)

When asked, go to https://classroom.github.com (while logged into GitHub) and wait for instructions.

Your First Assignment

A quick systems check using GitHub, Jupyter, and a Unix Terminal.

Git / GitHub Classroom Workflow



1. Fork a copy of the Deals repo.

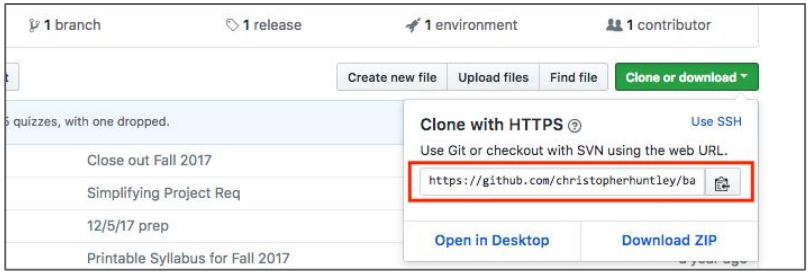
Click the assignment link on the class agenda. You will be directed to GitHub Classroom after indicating your account on the class roster.

GitHub will then create a forked copy of the assignment to your GitHub account.

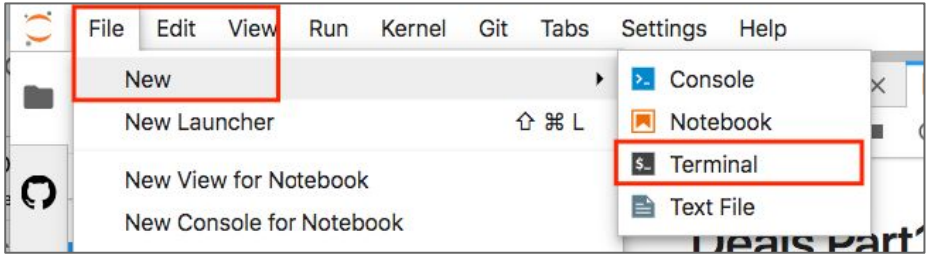
- A **fork** is a personal copy of a repository with you as the owner (so you can modify things). You do not have permission to edit the original copy.
- The fork needs to be **cloned** to a workspace in order for you to work on it. **We'll do that in the next step.**

2. Clone the Repo to JupyterLab.

On GitHub, get (copy) a clone URL for your forked repository.



In Jupyter Lab Launch a new Terminal tab.



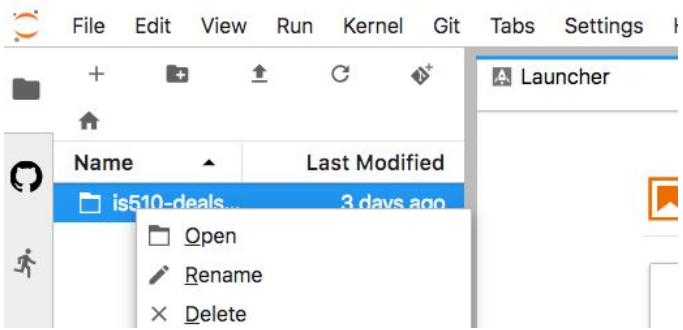
Then type (and paste)

```
git clone <your clone URL>
```

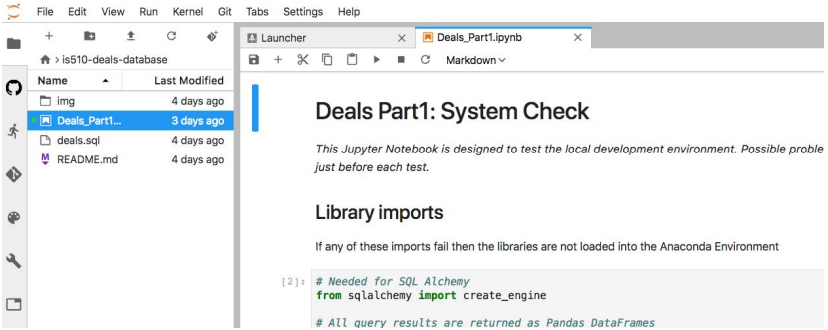
```
git clone https://github.com/christopherhuntley/ba5
```

3. Open the Deals_Part1 Notebook.

Open the repository folder in JupyterLab. The folder is your *local* copy of the repo. The one at GitHub is the *remote*.



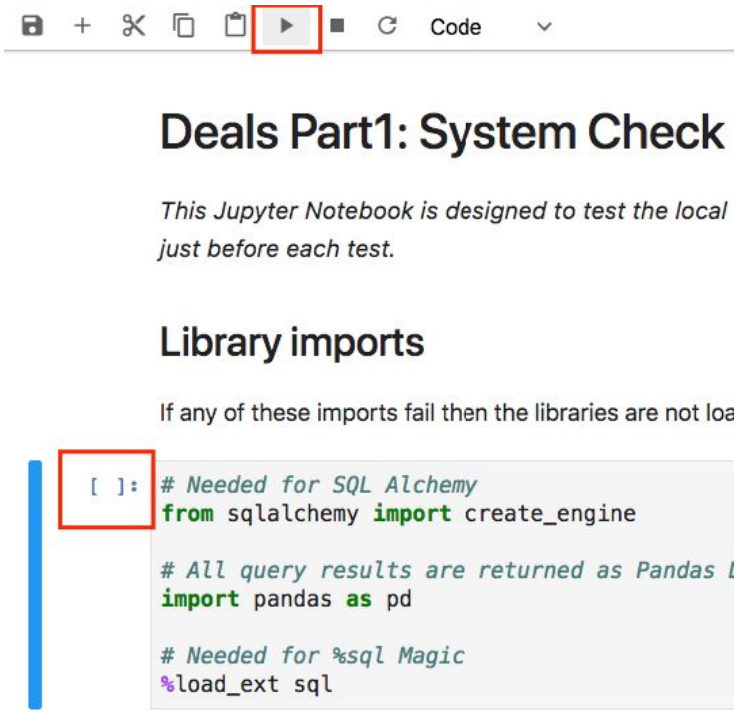
Open the Deals_Part1.ipynb file. It is a prewritten Jupyter Notebook with code to connect to a MySQL database.



4. Run A cell.

Jupyter Notebooks can be run one cell at a time or all at once. For now, let's try it cell-by-cell.

Click on the first cell with [ ] next to it. Then click the black triangle at the top to run it. After running, Jupyter will put a number in the [ ].



5. Complete the Test.

Step down the notebook, one cell at a time.

- Click the black triangle (or press Shift-Enter) to run a cell and advance to the next.
- Some cells will have Markdown text in them. Others will have Python or SQL code.
- This notebook tries the same query twice, first in Python, then in straight SQL.

SQL Alchemy Check

If this fails then either ...

- pymysql is missing (check in Anaconda) or
- the MySQL server is not running or misconfigured
- the deals database is not loaded into MySQL server

[2]:
```
# Initialize
engine = create_engine("mysql+pymysql://dealsuser:deals@localhost/deals")
with engine.connect() as conn, conn.begin():
    companies=pd.read_sql('Select * from Companies', conn)
companies
```

[2]:

	CompanyID	CompanyName
0	1	3Com Corporation
1	2	Abitibi-Price Inc.
2	3	ADT Limited

%sql Magic Check

If the following cells fail, then either ...

- The database connection failed or
- Pandas is not imported or
- The wrong version of ipython-sql is installed; want v0.3.8 or later

[3]:
```
# Connect to the deals database
%sql mysql+pymysql://dealsuser:deals@localhost/deals
```

[3]:
```
'Connected: dealsuser@deals'
```

[4]:
```
# %sql magic returns a ResultSet that can be converted t
result = %sql select * from Companies
companies = result.DataFrame()
companies
```

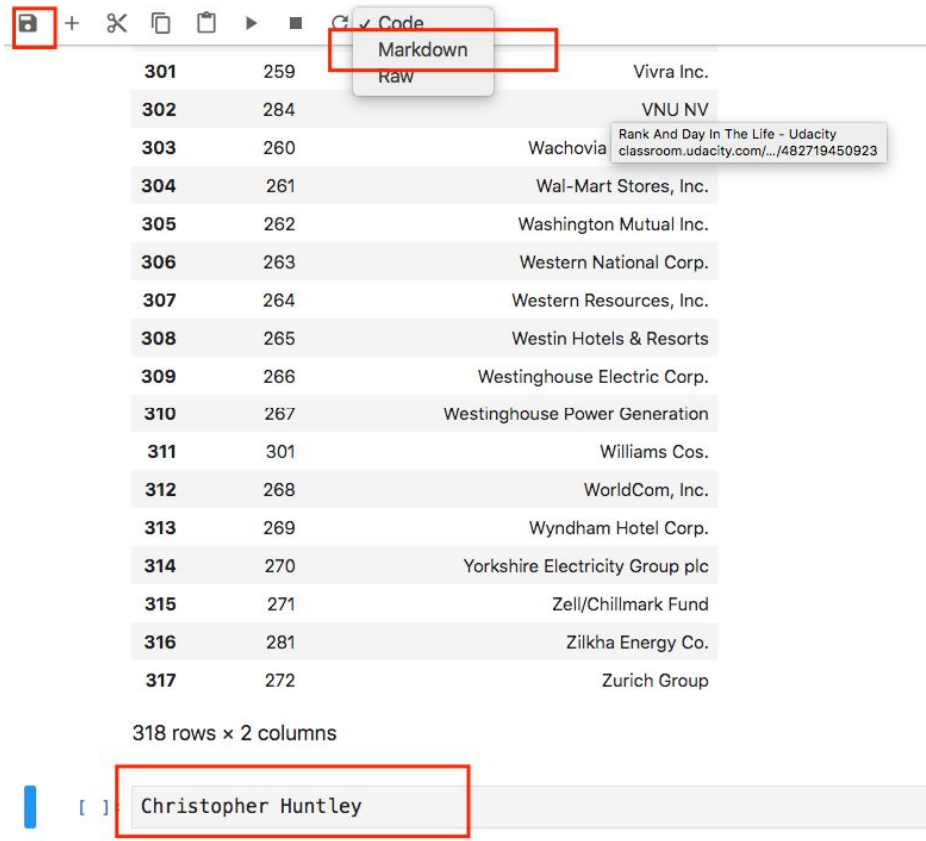
```
 * mysql+pymysql://dealsuser:***@localhost/deals
318 rows affected.
```

[4]:

	CompanyID	CompanyName
0	1	3Com Corporation
1	2	Abitibi-Price Inc.
2	3	ADT Limited
3	4	Advanta Corp. Credit card unit
4	5	AirTouch Communications, Inc.

6. Sign Your Work.

1. Write your name in the last (empty cell).
2. Change the cell type from Code to Markdown.
3. Save the Notebook.



301	259	Vivra Inc.
302	284	VNU NV
303	260	Wachovia
304	261	Wal-Mart Stores, Inc.
305	262	Washington Mutual Inc.
306	263	Western National Corp.
307	264	Western Resources, Inc.
308	265	Westin Hotels & Resorts
309	266	Westinghouse Electric Corp.
310	267	Westinghouse Power Generation
311	301	Williams Cos.
312	268	WorldCom, Inc.
313	269	Wyndham Hotel Corp.
314	270	Yorkshire Electricity Group plc
315	271	Zell/Chillmark Fund
316	281	Zilkha Energy Co.
317	272	Zurich Group

318 rows × 2 columns

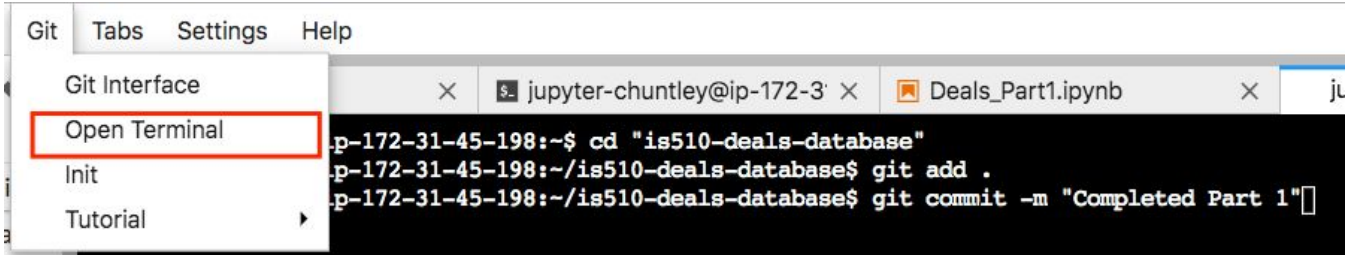
7. Commit your repository changes.

1. Close the open Terminal tab.
2. From the Git menu select "Open Terminal" to get a new Terminal within your repository directory.
3. Type

```
git add .
git commit -m "Completed Part 1"
```

to log your changes to the files.

You will be asked for contact info. Ask your classmates for help.



8. Push your work back to GitHub.

Your local git repo is up-to-date, but GitHub isn't.

1. In the Terminal type

   ```
   git push
   ```

2. Then check to make sure your changes pushed to GitHub. The log message "Completed Part 1" should appear next to the `Deals_Part1.ipynb` file in your GitHub repo. If you open the notebook it should have your name at the bottom.

You will be asked for GitHub account info. Ask your classmates for help.

9. Shut Down Jupyter Lab

JupyterLab is a shared resource. CPU time is expensive and idle kernels affect everybody else.

Please shut down your workspace when you are not using it.

1. From the Kernel menu select "Shut down all kernels".
2. Log out from Jupyter Lab.

Fairfield University
DOLAN
School of Business

Databases for Analytics

Course Onboarding

Syllabus, Software, Tutorials, etc.

Old School Desktop Software Installation

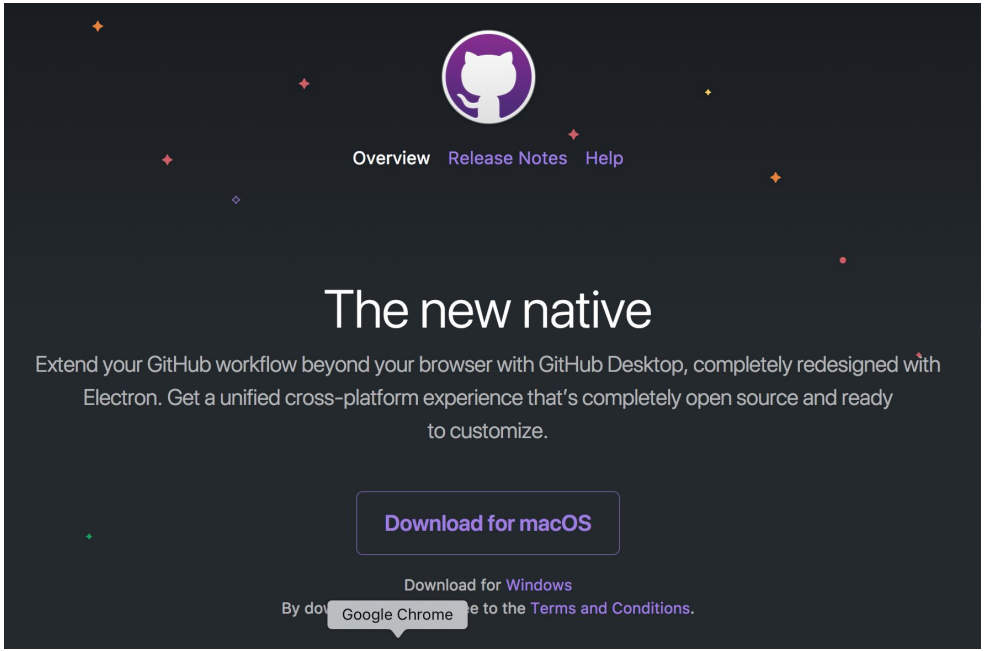
For those of you who need to work offline
(or if BA Lab is not working)

The following follows includes a "systems check"
like we did in class.

Install GitHub Desktop

Download from desktop.github.com.

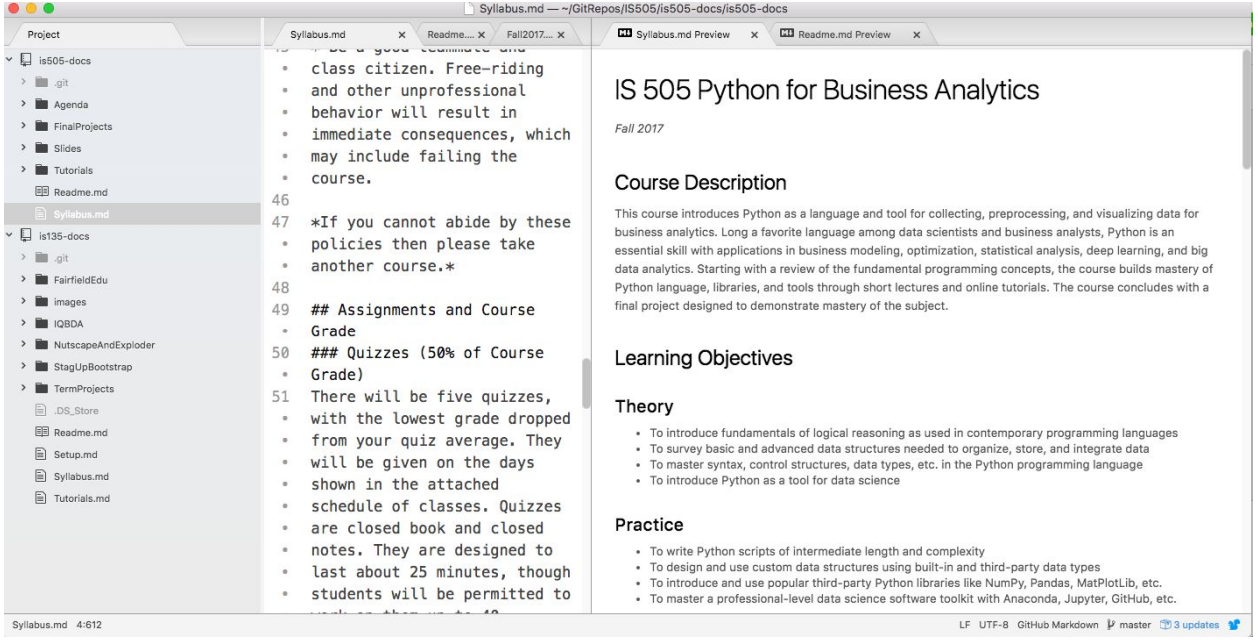
Then install as usual.



Install Atom (Recommended)

A code editor that works great with GitHub.

Install from atom.io



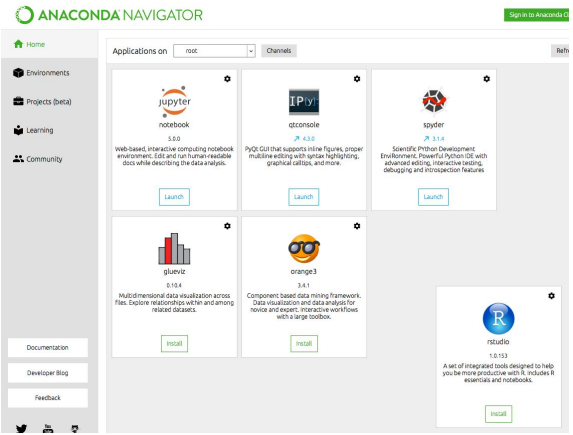
Create a Folder for your work

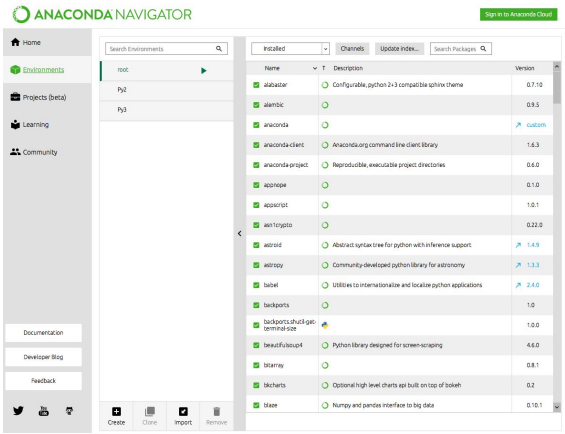
1. Create a new folder called **`BA510`** in your **`documents`** folder or desktop.
2. All your local Git repositories and other work will be in this new **`BA510`** folder.
3. Take note of where you created the folder. You will need it later.

Anaconda

Anaconda is a desktop Python environment that bundles lots of tools and packages:

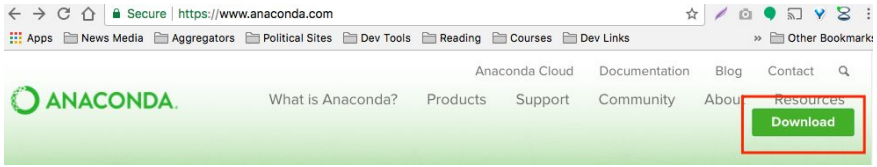
- Python (Installation)
- Apps: Jupyter Notebooks, Spyder IDE, etc.
- Libraries: NumPy, MatPlotLib, etc.
- Conda: command line tools





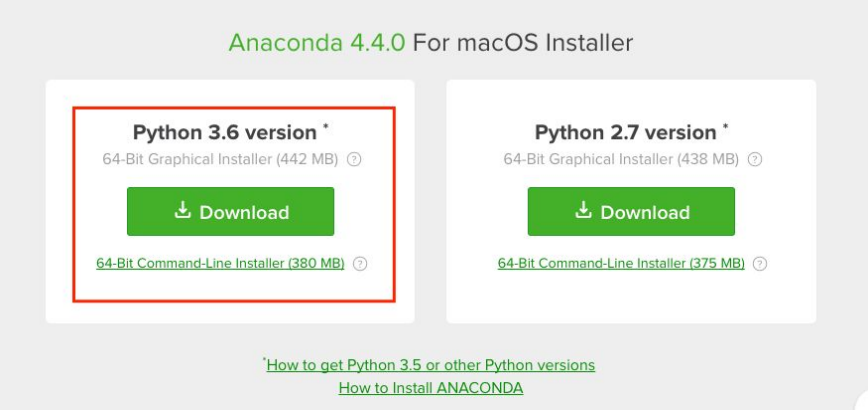
Install Anaconda

Go to [anaconda.com](https://www.anaconda.com) and click the download button.



Choose the Python 3.* version for your OS.

The download may take a while. Be patient.

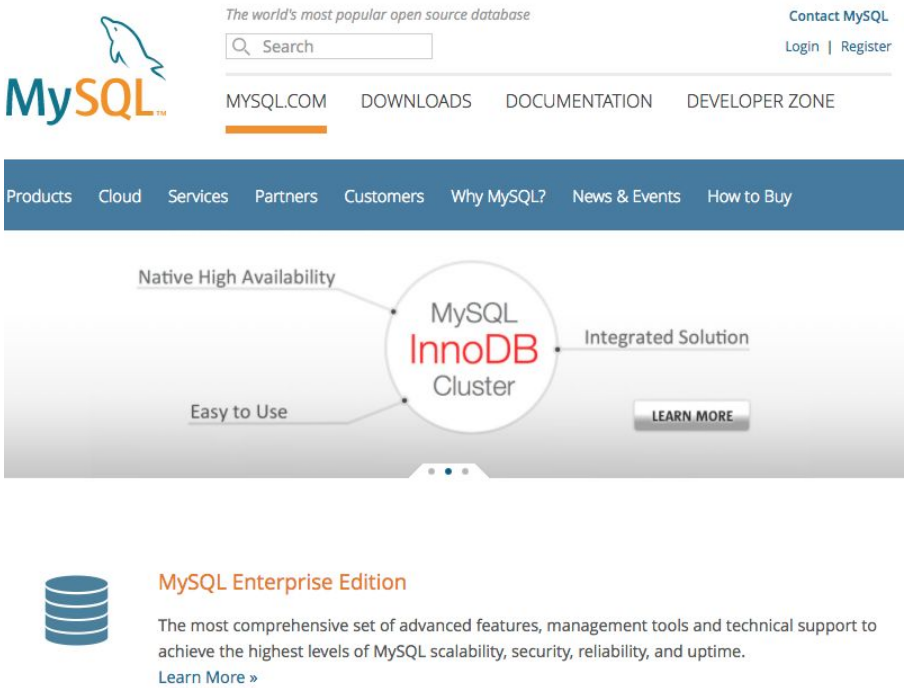


Install as usual.

Install MySQL

MySQL is Oracle's open source DBMS. It is widely used for web apps.

We need both **MySQL Server** and **MySQL Workbench**.



Windows vs MacOS

How to install MySQL depends on your operating system.

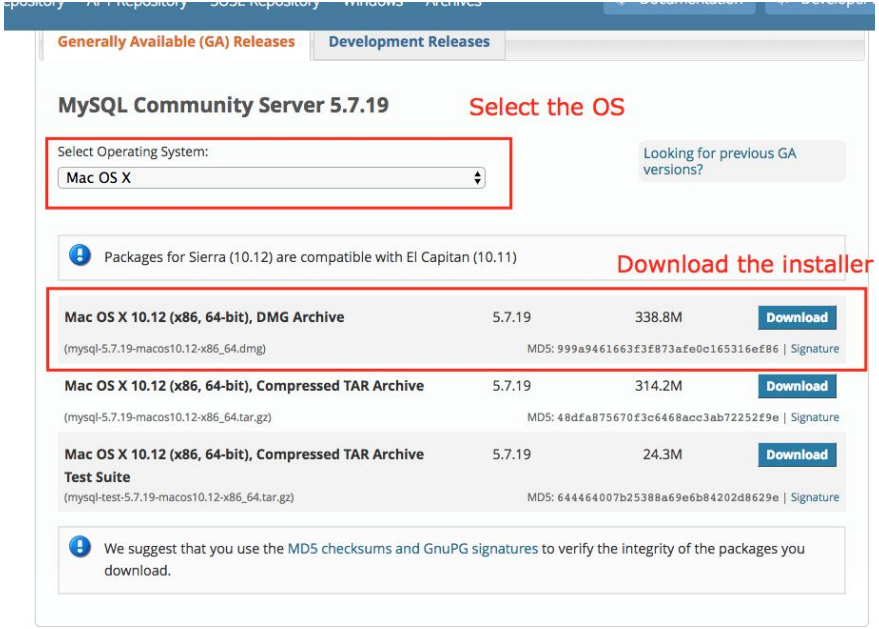
Windows: Use the all-in-one installer.

MacOS: Install **MySQL Server (v5.7)** and then **MySQL Workbench (v6.1)** to work around a bug in Mac OS 10.13 (High Sierra).

MacOS: MySQL Server Community Edition

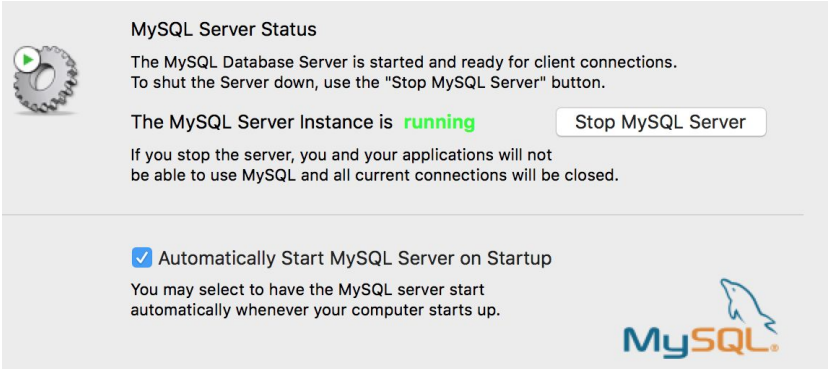
Download and install the latest release in the v5.7 series.

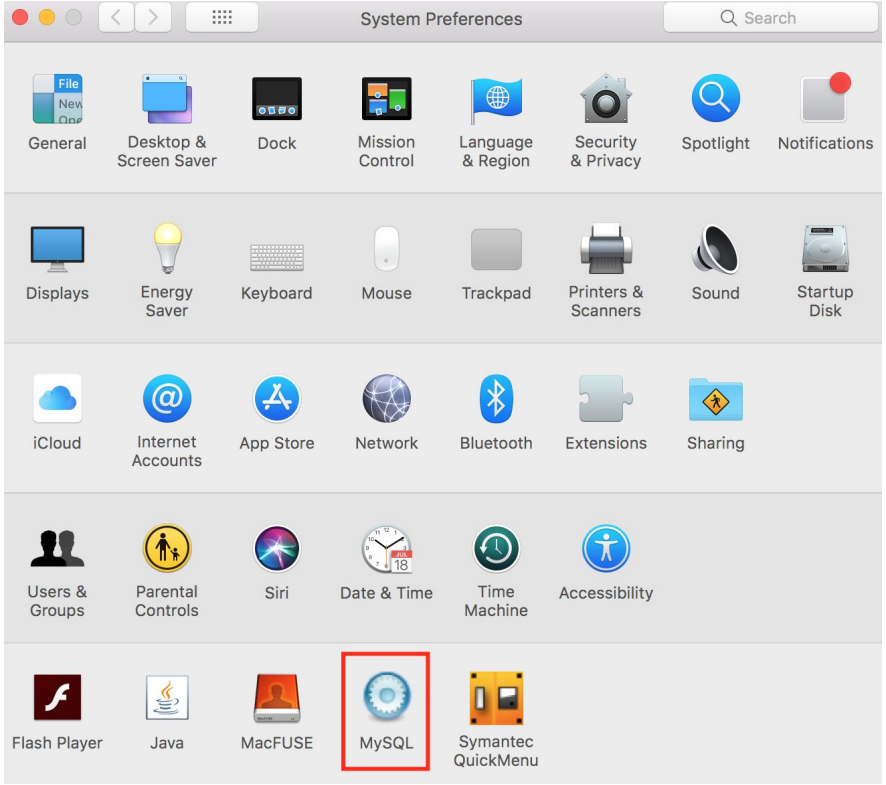
After installing, you will need to reboot to get the MySQL launcher in your preferences panel.



MacOS: Check for MySQL launcher

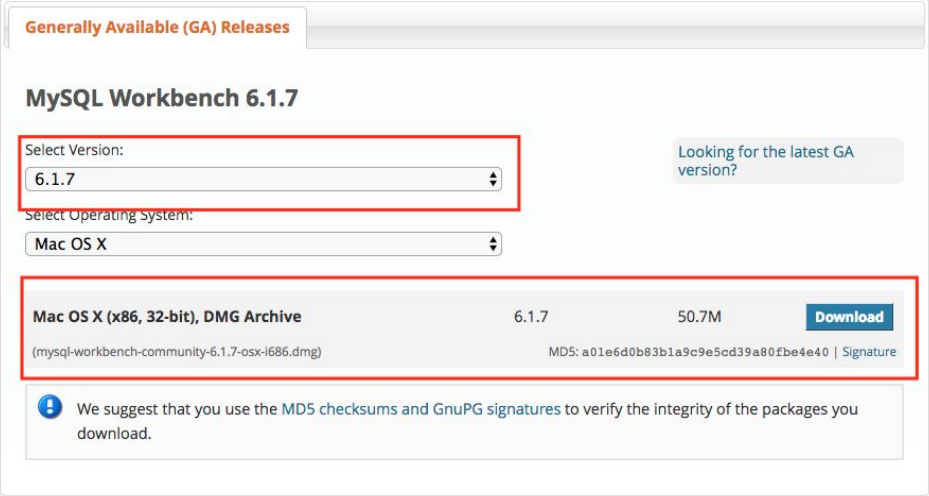
MySQL should show up in your **System Preferences** panel. Click to start/stop the server.





MacOS: Install MySQL Workbench

MySQL Workbench is an app for managing and querying MySQL Databases.



Install version 6.1.7. Later version are not yet compatible with MacOS 10.13

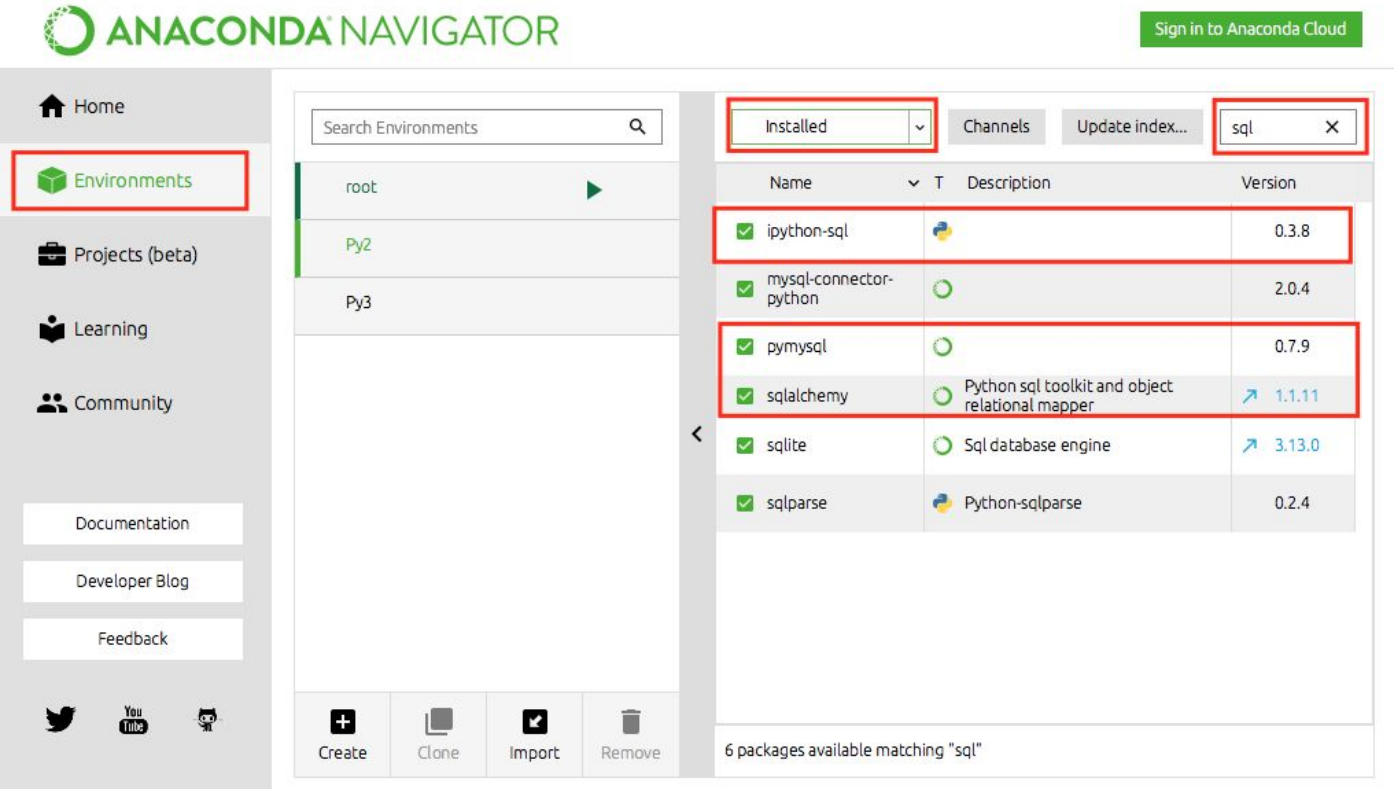
Anaconda Add-ons

Anaconda is missing a few things we'll want in order to connect Jupyter to our databases. We'll need to …

1. Make sure **`sqlalchemy`** is installed/enabled
2. Install the **`pymysql`** bridge library
3. Install the **`ipython-sql`** magic for Jupyter

We will use a combination of Anaconda Navigator and the command line.

Anaconda Environment/Packages



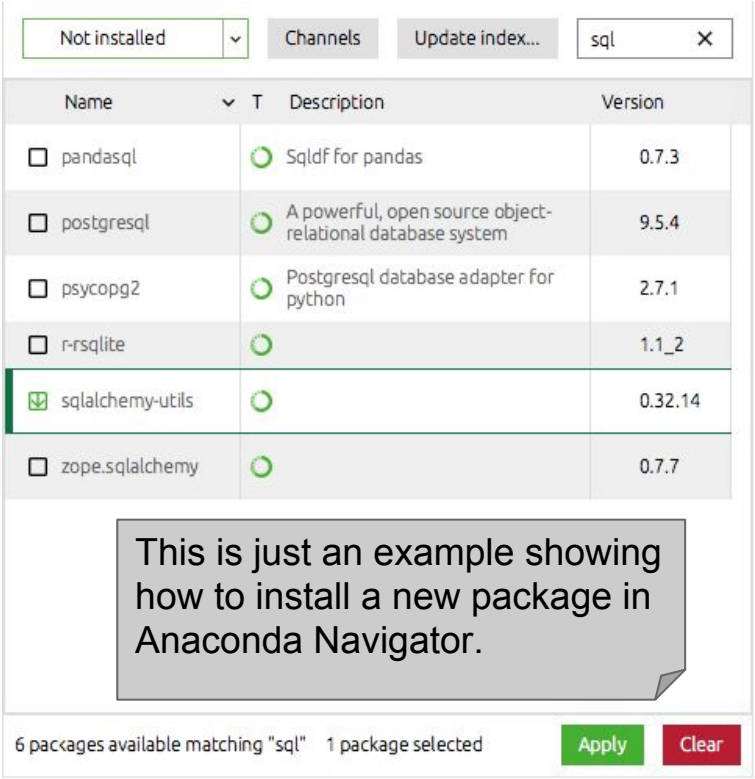
A complete Installation looks like this.

We'll do it one step at a time.

SQL Alchemy

SQL Alchemy provides a bunch of useful Python utilities.

1. Check for SQL Alchemy in your Installed packages for the root environment.
2. If it is not installed then install it from the Not Installed packages list.



PyMySQL Package

PyMySQL is a Python driver for connecting to MySQL databases.

1. Open the Command Prompt (Windows)/ Terminal (MacOS).
2. Use the `conda` package manager to find and install the package.

```
conda install -c anaconda pymysql
```

chuntley — conda install -c anaconda pymysql — 80×24

```
Last login: Wed Oct 11 22:08:21 on ttys003
DSB1122-C4148M:~ chuntley$ conda install -c anaconda pymysql
Fetching package metadata ...............
Solving package specifications: .

Package plan for installation in environment /Users/chuntley/anaconda:

The following packages will be UPDATED:

    conda:     4.3.27-py36hb556a21_0 --> 4.3.30-py36h173c244_0 anaconda
    pymysql:   0.7.9-py36_0          --> 0.7.11-py36h75d80ff_0 anaconda

The following packages will be SUPERSEDED by a higher-priority channel:

    conda-env: 2.6.0-0

Proceed ([y]/n)?
```

This is the MacOS Terminal, but it looks similar in the Windows Command Prompt

`ipython-sql` Package

This adds special sql "magic" for Jupyter Notebooks.

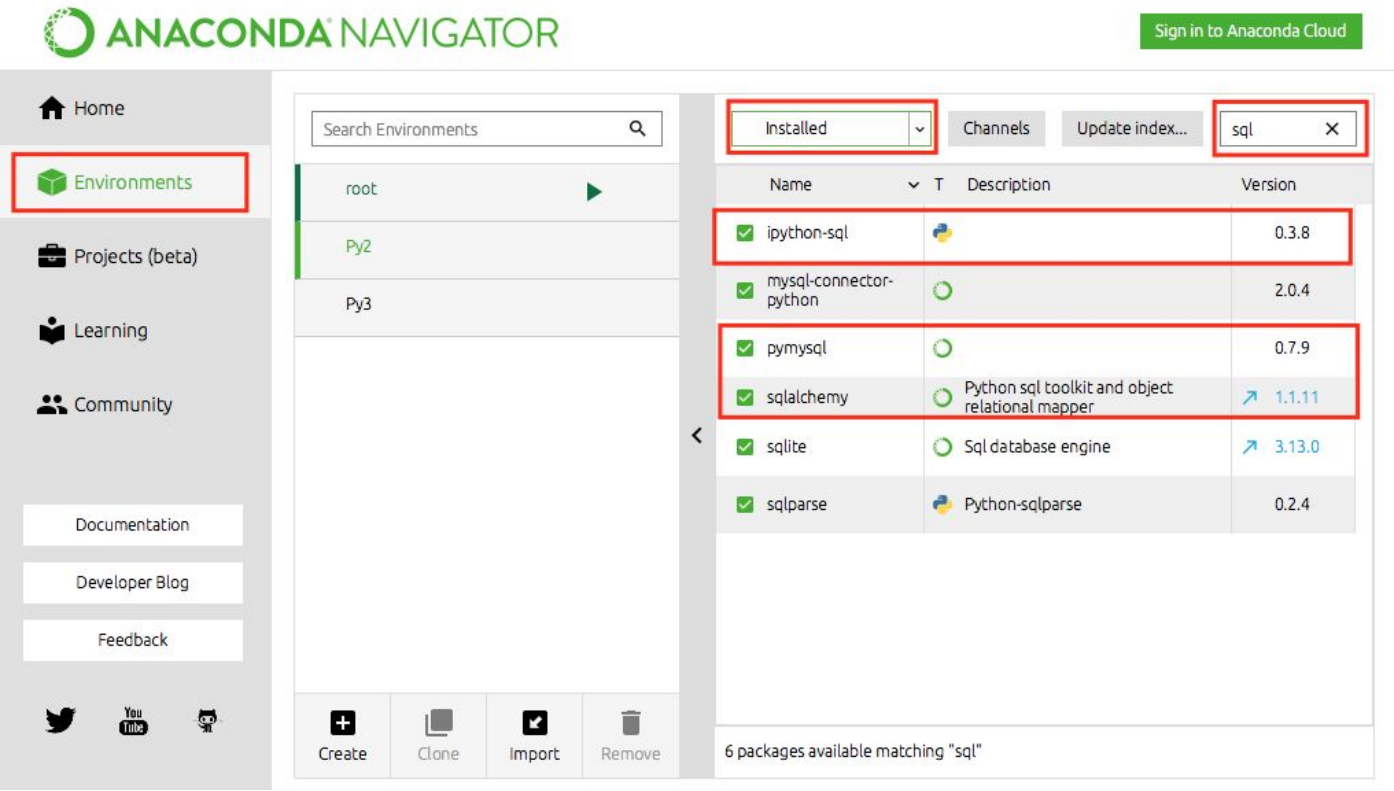
1. Install from the command line.
2. Use `pip` as the package manager.

```
pip install ipython-sql
```

chuntley — -bash — 69×40

```
DSB1122-C4148M:~ chuntley$ pip install ipython-sql
Collecting ipython-sql
  Downloading ipython_sql-0.3.8-py3-none-any.whl
Collecting sqlparse (from ipython-sql)
  Downloading sqlparse-0.2.4-py2.py3-none-any.whl
Requirement already satisfied: sqlalchemy>=0.6.7 in ./anaconda/lib/py
thon3.6/site-packages (from ipython-sql)
Requirement already satisfied: six in ./anaconda/lib/python3.6/site-p
ackages (from ipython-sql)
Collecting prettytable (from ipython-sql)
  Downloading prettytable-0.7.2.zip
Requirement already satisfied: ipython>=1.0 in ./anaconda/lib/python3
.6/site-packages (from ipython-sql)
Requirement already satisfied: ipython-genutils>=0.1.0 in ./anaconda/
lib/python3.6/site-packages (from ipython-sql)
Building wheels for collected packages: prettytable
  Running setup.py bdist_wheel for prettytable ... done
  Stored in directory: /Users/chuntley/Library/Caches/pip/wheels/b6/9
0/7b/1c22b89217d0eba6d5f406e562365ebee804f0d4595b2bdbcd
Successfully built prettytable
Installing collected packages: sqlparse, prettytable, ipython-sql
Successfully installed ipython-sql-0.3.8 prettytable-0.7.2 sqlparse-0
.2.4
```

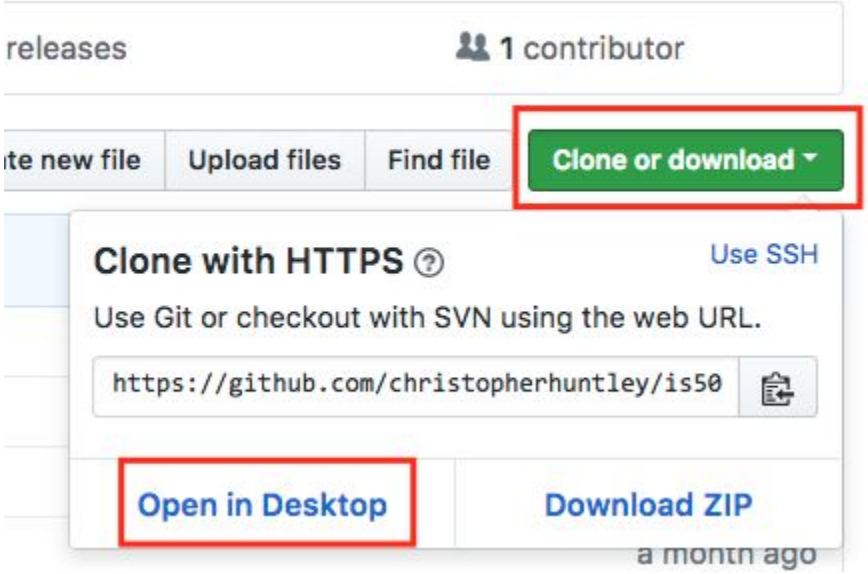
Anaconda Navigator Again



GitHub Desktop Check

Clone your forked copy of the repository to your desktop.

Save the repository in your new **`BA510`** folder.

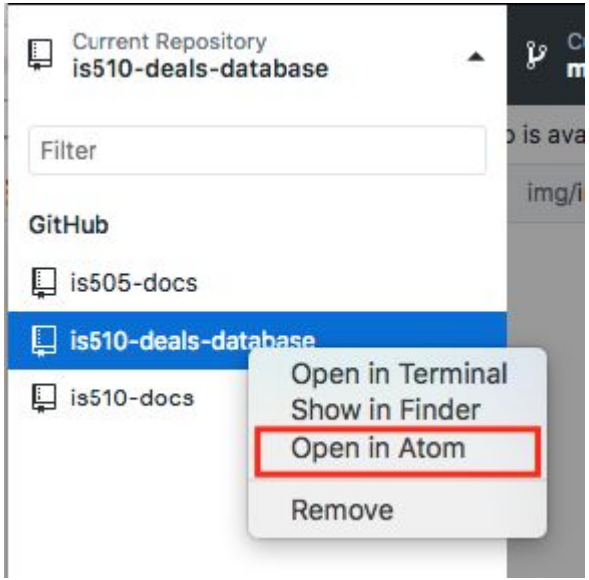


Atom Editor Check

From the repository pane, right click on the repository and select Open in Atom.

Atom should appear with the repository contents listed on the left.

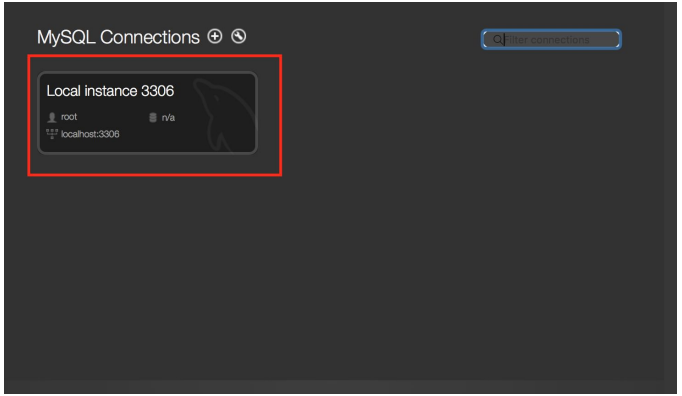
Open the **`deals.sql`** file.

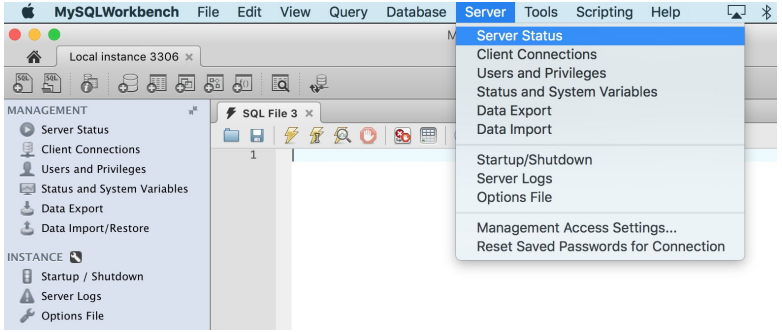


MySQL Server Check

MySQL Workbench Check

1. Open MySQL Workbench
2. Choose your running instance of MySQL Server
3. Check that MySQL Workbench can control the server.

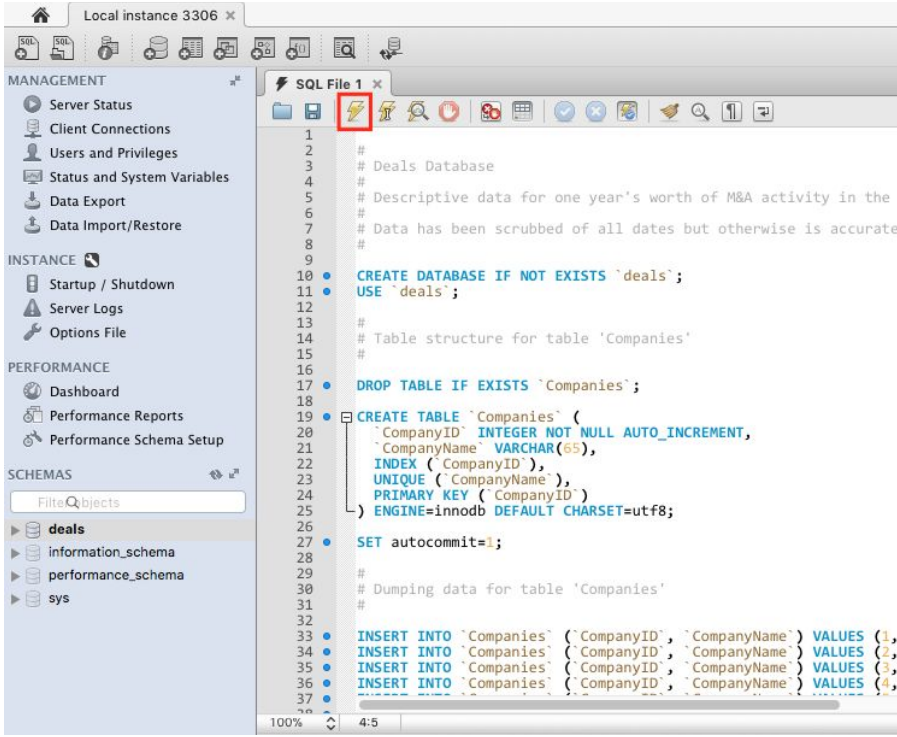




Loading the Database

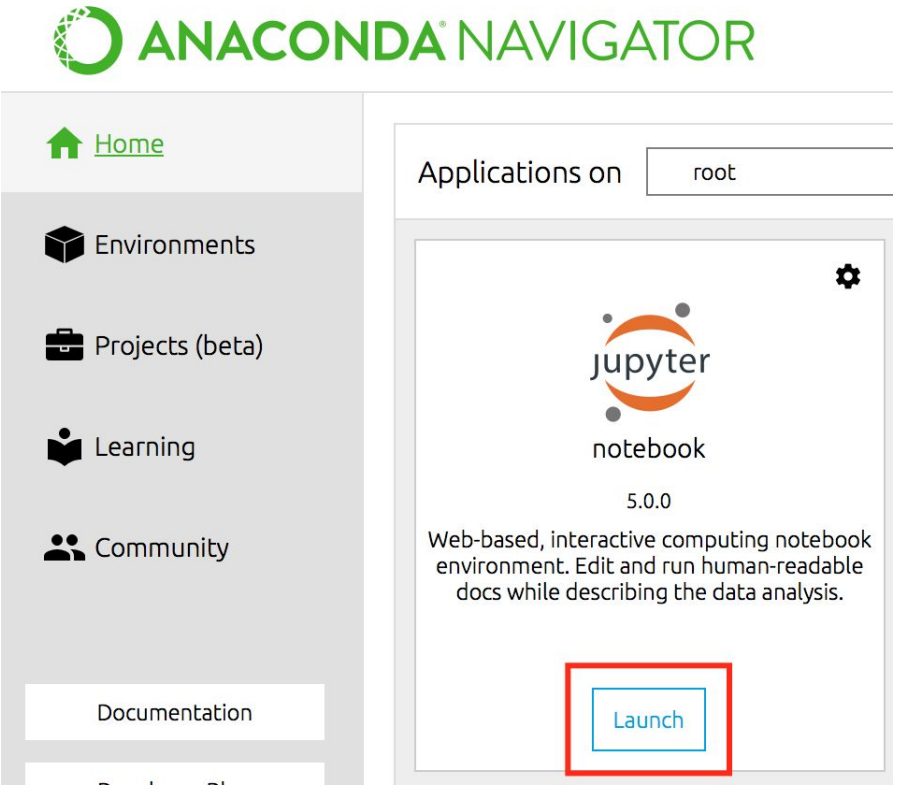
Run the **`deals.sql`** script:

1. `File → Open SQL Script …`
2. Navigate to your repository folder.
3. Select the `deals.sql` file.
4. Click the lightning bolt icon to run the script.
5. The `deals` schema should appear in the left panel.



Jupyter Notebook Check

1. Open Anaconda Navigator (if not open).
2. Launch Jupyter Notebook.
3. Open the **`Deals_Part1.ipynb`** notebook in your repository folder.



SQL Alchemy, PyMySQL Check

The first part of the notebook sets up a connection to the database, much like we just did with MySQL Workbench. This is where the PyMySQL Package comes into play.

Run the first cell to check if PyMySQL is working correctly. You should get a table of company names.

`%sql` Magic Check

The next code cell uses **`%sql`** 'magic' to embed SQL code directly into a Python assignment statement.

Run the cell. The variable **`companies`** is a Pandas DataFrame which is displayed as a table.

If this doesn't work just like the first code cell then the ipython-sql package is not installed correctly.

Sign your work

Add a new Markdown cell with your name in it to the bottom of the notebook.

Save the notebook.

Sync to GitHub

In GitHub Desktop, note that Git has detected your edit to the notebook.

- **Commit** your changes with the comment "Completed Part 1"
- **Push** your updated repository to GitHub