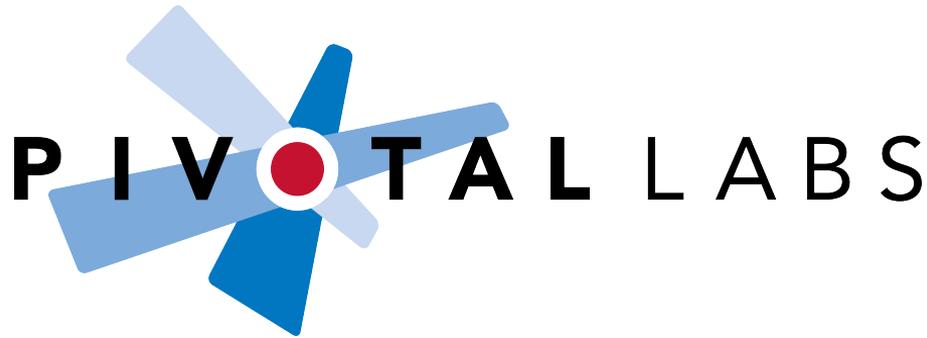
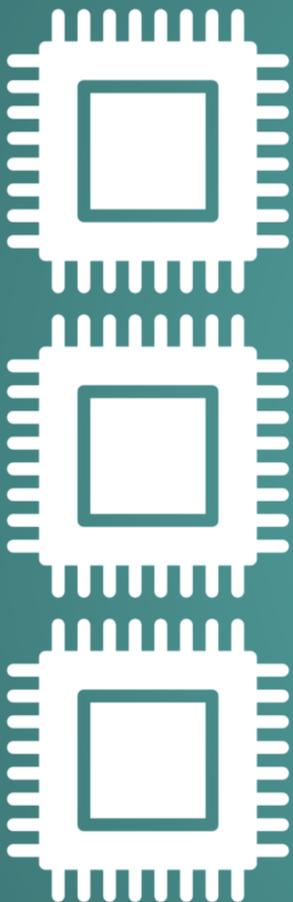


# Bring Your Code to Your Data

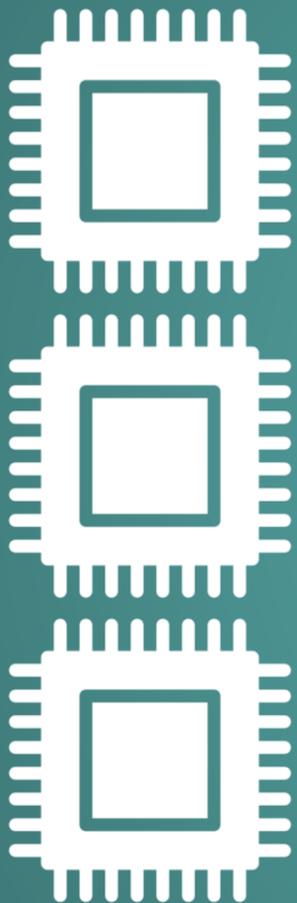
**Ian Huston**  
**@ianhuston**

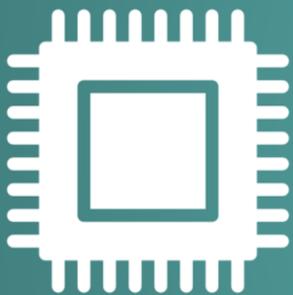




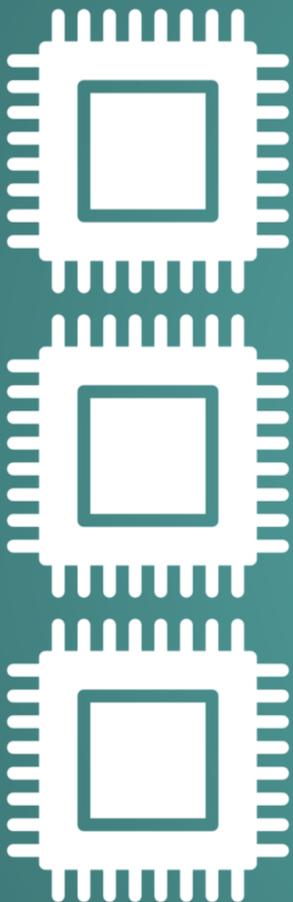
# High Performance Computing

HPC  $\neq$  BIG DATA

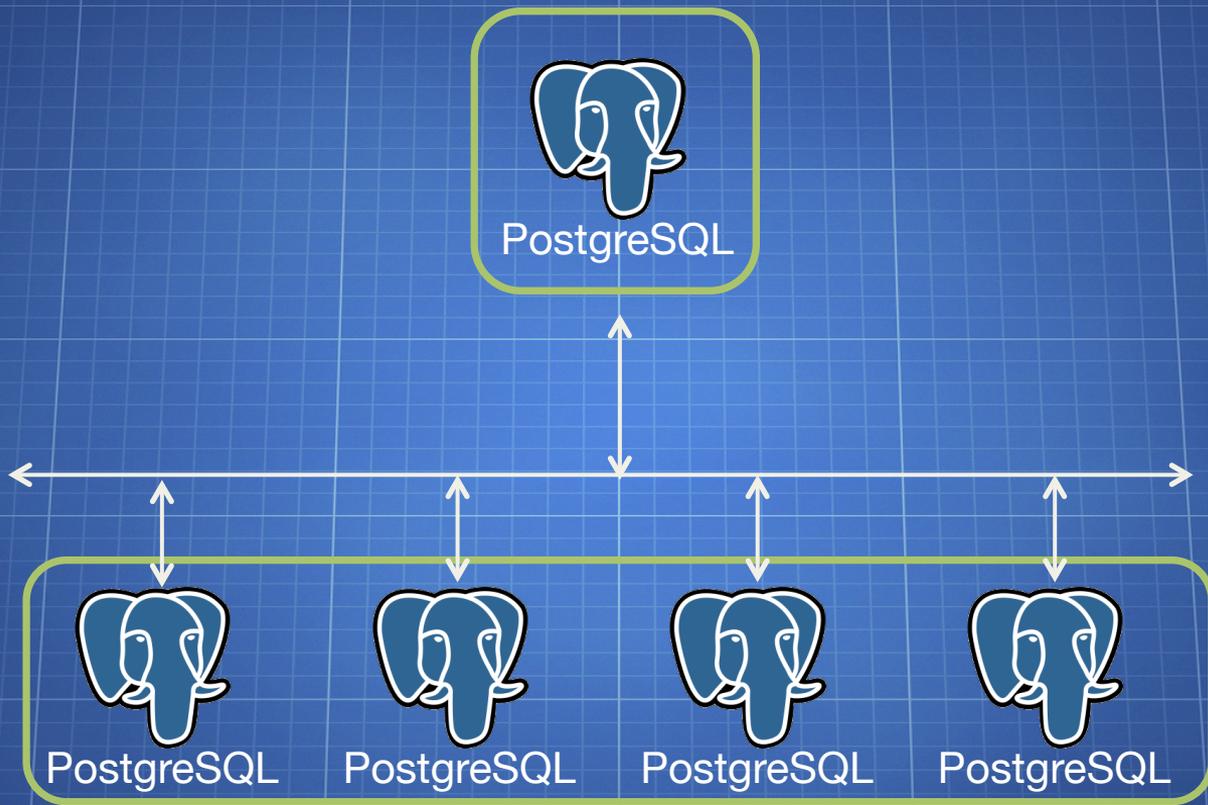




# In-Database Computing



# In-Database Distributed Computing





- 1. SQL for analytics**
- 2. Packaged libraries**
- 3. In-Database Python**



# 1. SQL for analytics

# SQL is more than SELECT

- Window Functions
- WITH queries (Common Table Expressions)
- User Defined Aggregations
- User Defined (SQL) Functions

Examples:

- Time Series using Windowing: <http://blog.pivotal.io/author/caleb-welton>
- Heroku's Postgres bits: <http://postgres-bits.herokuapp.com>

# Get rows in 6th decile of scores

CTE

Window Function

Normal Query

```
WITH decile as (  
  SELECT *,  
         ntile(10) OVER ( ORDER BY score )  
  FROM mytable)  
SELECT *  
FROM decile  
where ntile = 6;
```

<http://postgres-bits.herokuapp.com/#60>

# User Defined (SQL) Functions

Declaration { CREATE FUNCTION times2(INT)  
RETURNS INT  
AS \$\$  
SQL { SELECT 2 \* \$1  
Statements { \$\$  
Language { LANGUAGE sql;  
Execution { SELECT times2(1);  
times2  
-----  
2



## 2. Packaged libraries

# IN-DATABASE MACHINE LEARNING

MADlib

<http://madlib.net>

# IN-DATABASE GEOGRAPHIC QUERIES



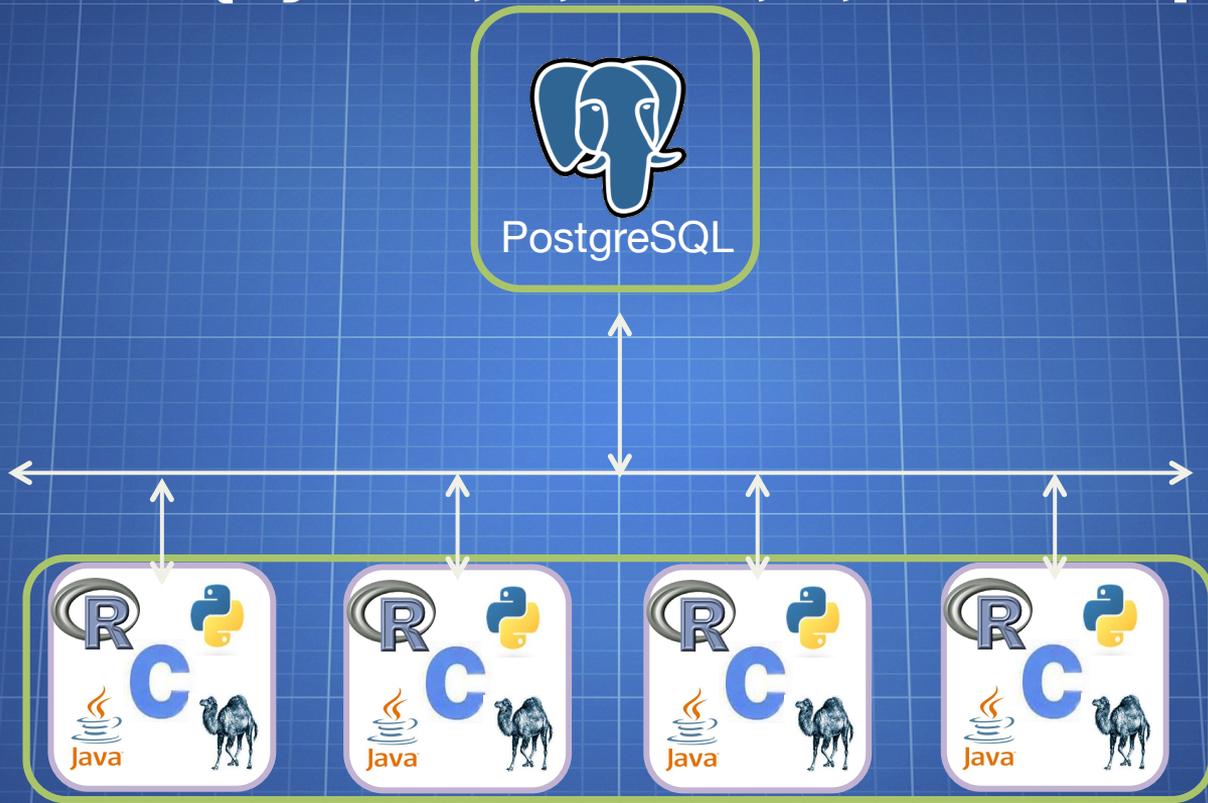
<http://postgis.net>

Pivotal



### **3. In-Database Python (and R, Java, C, etc)**

# PL/X : X in {Python, R, Java, C, JavaScript, etc.}



# Data Parallelism

- Little or no effort is required to break up the problem into a number of parallel tasks, and there exists no dependency (or communication) between those parallel tasks.
- Examples:
  - Measure the height of each student in a classroom (explicitly parallelizable by student)
  - MapReduce
  - `map()` function in Python

```
SQL wrapper { CREATE FUNCTION
                pymax (a integer, b integer)
                RETURNS integer
                AS $$
                if a > b:
                return a
                return b
Normal Python {
Language { $$ LANGUAGE plpythonu;
```

# **BENEFITS:**

**Reuse Python & R code**

**Access Python & R libraries**

**Implicit parallelism**

# Financial Examples



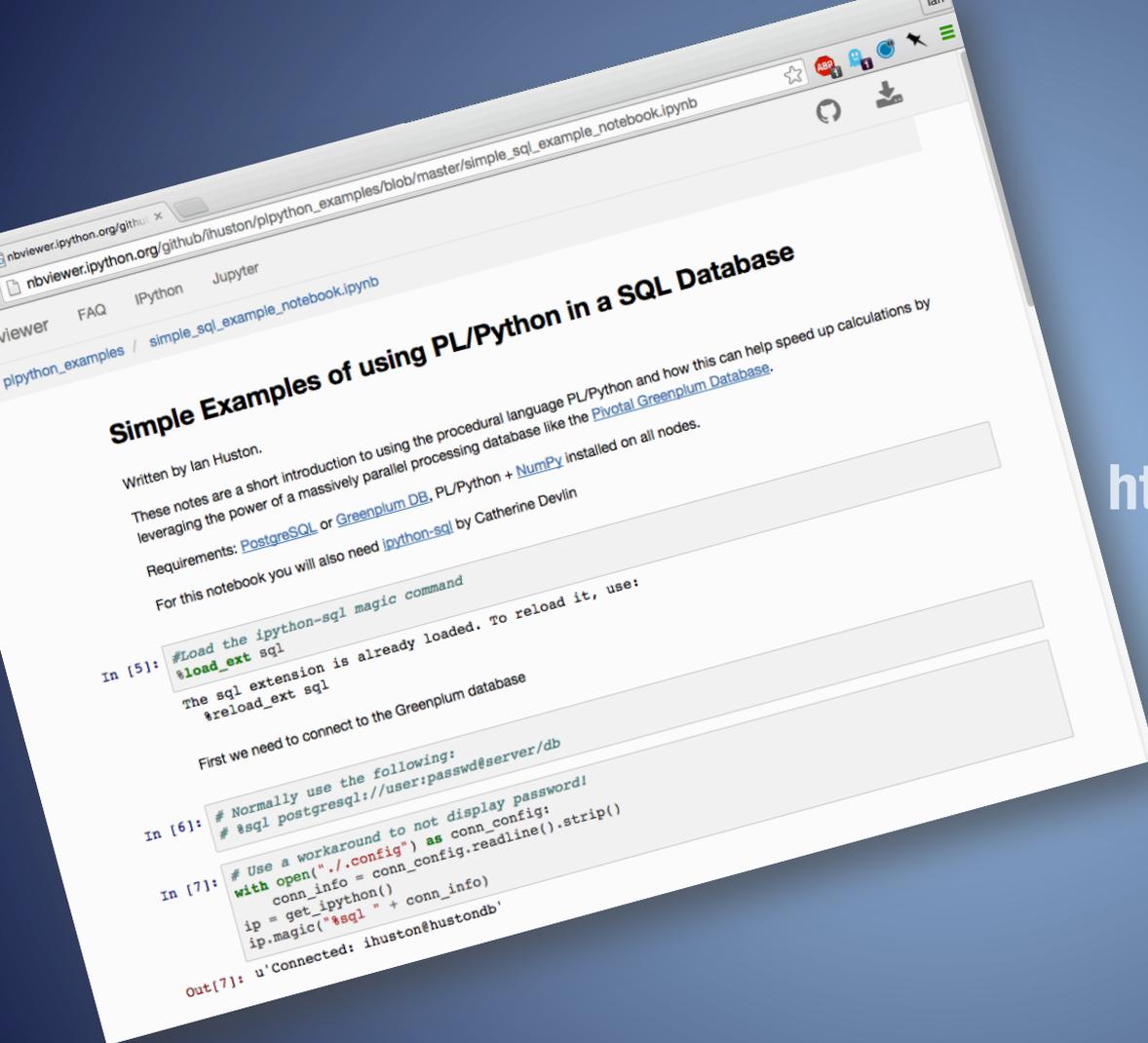
Insurance Risk Analysis



Stress Testing



Asset Management  
and Churn



## Simple Examples of using PL/Python in a SQL Database

Written by Ian Huston.

These notes are a short introduction to using the procedural language PL/Python and how this can help speed up calculations by leveraging the power of a massively parallel processing database like the [Pivotal Greenplum Database](#).

Requirements: [PostgreSQL](#) or [Greenplum DB](#), PL/Python + [NumPy](#) installed on all nodes.

For this notebook you will also need [ipython-sql](#) by Catherine Devlin

```
In [5]: #Load the ipython-sql magic command
%load_ext sql
```

The sql extension is already loaded. To reload it, use:  
`%reload_ext sql`

First we need to connect to the Greenplum database

```
In [6]: # Normally use the following:
# %sql postgresql://user:passwd@server/db
```

```
In [7]: # Use a workaround to not display password!
with open("../config") as conn_config:
    conn_info = conn_config.readline().strip()
ip = get_ipython()
ip.magic("%sql " + conn_info)
```

```
Out[7]: u'Connected: ihuston@hustondb'
```

# MORE DETAILS

<http://tinyurl.com/ih-plpython>

@ianhuston

