**PostgreSQL Syntax Basics**

First, you need to know that you need access to specific databases and that separate databases cannot communicate with each other (like postgis\_1 and postgis\_2) while others can (anything data tables inside of postgis\_1 or postgis\_2). Additionally, you need to know that tables usually have primary keys, which is a field in a table that uniquely identifies each row in a table. Tables can only have one primary key, but that key can be multiple fields. Primary keys are important because they identify the uniqueness of each row in the table.

For our lab, you only need to know a specific subset of PostgreSQL commands, which are similar to the same commands that are available in the tidy verse package in R. Technically, you could do all of the selecting and filtering in tidy verse, but then you have to spend the time bringing the entire table into R, which is not ideal when you have millions of rows or features. Thus, you need to go into the table you want to draw from and click on “SQL command” (on the right) and use the following commands to select and filter based on important attributes:

-- = # in R (ignores command)

SELECT ([link](https://www.postgresql.org/docs/9.5/sql-select.html)) = Selects only certain variables

SELECT \* = Shortcut for selecting all variables

FROM = From which dataset

LIMIT = Useful to start with because it only selects a specific subset of dataset

WHERE = Equivalent to filter in tidy verse

HAVING = When a condition is met (like WHERE)

GROUP BY = Equivalent to group\_by in tidy verse

ORDER BY = Puts column in order

JOINS: (INNER JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN) “ON (Var1 = Var2)” ([link](https://www.postgresql.org/docs/8.3/tutorial-join.html))

[**SELECT**](https://www.postgresql.org/docs/11/static/sql-select.html) author\_id, login   
FROM [project\_author\_commits\_full](http://sdad.policy-analytics.net:8080/?pgsql=postgis_2&username=kb7hp&db=ghtorrent_restore&ns=public&table=project_author_commits_full)   
WHERE forked\_from IS NULL

Alternatively, you can just import the data into R and clean it with the typical tidyverse functions using the following procedure. First, create an .Renviron file by creating a new R Script file and naming it “.Renviron” (accepting the error note that comes after) and including similar information to this in the file:

```{r}

R\_LIBS=/home/kb7hp/.rpackages

db\_userid = "kb7hp"

db\_pwd = " "

```

Once you save this file, restart R so it acknowledges the .Renviron file.

Then, use this code to pull from the correct place:

```{r}

conn <- dbConnect(drv = PostgreSQL(),

dbname = "ghtorrent\_restore", # Name of the highest level database

host = "postgis\_2", # Name of the lower level database

port = 5432L,

user = Sys.getenv("db\_userid"), # This pulls from the .Renviron file

password = Sys.getenv("db\_pwd")) # This pulls from the .Renviron file

data <- dbReadTable(conn = conn, name = c(schema = "public", name ="project\_author\_commits\_full"))

```

Alternatively, you can all use a combination of the dbConnect + dbGetQuery functions to write the SQL code right into R and pull the sample of your data like that. Here is an example:

```{r}

conn <- dbConnect(drv = PostgreSQL(),

dbname = "ghtorrent\_restore", # Name of the highest level database

host = "postgis\_2", # Name of the lower level database

port = 5432L,

user = Sys.getenv("db\_userid"), # This pulls from the .Renviron file

password = Sys.getenv("db\_pwd")) # This pulls from the .Renviron file

oss\_sample <- dbGetQuery(conn, "SELECT author\_id, login, commit\_cnt, repo\_slug

FROM project\_author\_commits\_full

WHERE forked\_from IS NULL AND project\_deleted = FALSE

LIMIT 50") # you will likely want to start at 50 and increase slightly from there

```