



# ESTABLISHMENT OF REGIS HEALTHCARE INFORMATICS DATABASE

BY GILBERT ANTHONY BERNAL

# OUTLINE



Introduction



Database  
Requirements  
& Goals



Tools



Database  
Creation



Code Review



Analysis



Conclusion



Possible  
Projects

# INTRODUCTION

- Data Engineering project focus
- Project Goal: Design a database for the Regis Health Informatics school to use.
- Data used by health informatics school is from various health organization sources
- Data comes in various formats
  - Most common format is a CSV



National Institutes  
of Health



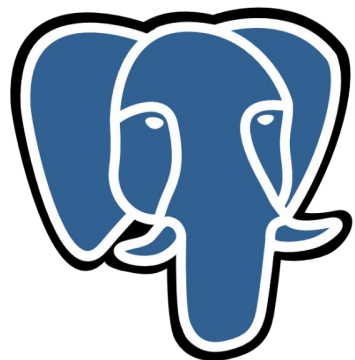
# DATABASE REQUIREMENTS & GOAL

## Requirements

- Database should be able to store any individuals health care data
- Must be able to create tables based off CSV file
- Must be able to load all data from CSV into its unique file

## Goal

- Create a database and code that will be able to pull all CSV files in a directory, create unique tables based off the CSV file headers, and store the data from these files into its unique table



PostgreSQL



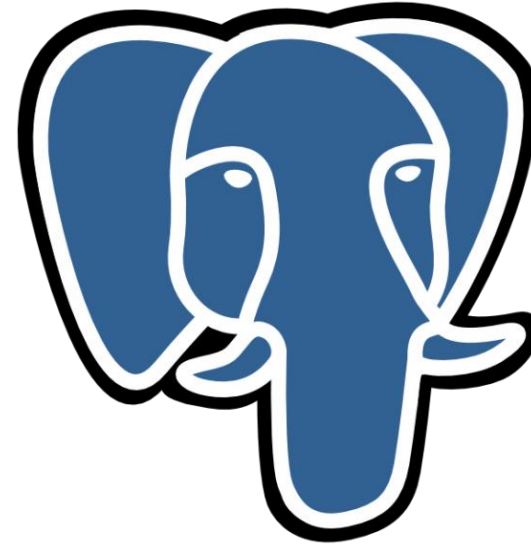
python



TOOLS

# POSTGRESQL

- Open source relational database system
- Uses standard SQL language
- Works with various datatypes
  - CSV, JSON, and XML
- Works with various coding languages
  - Python and Java
- Has method called COPY for uploading data files quickly



PostgreSQL

# PYTHON

- Open source high level programming language
- Various libraries to manipulate data and files
- Works well with PostgreSQL
- Has PYCOPG2 library
  - PYCOPG2 library is the most popular PostgreSQL adaptor for Python
  - Needed in order to use the COPY command for data upload



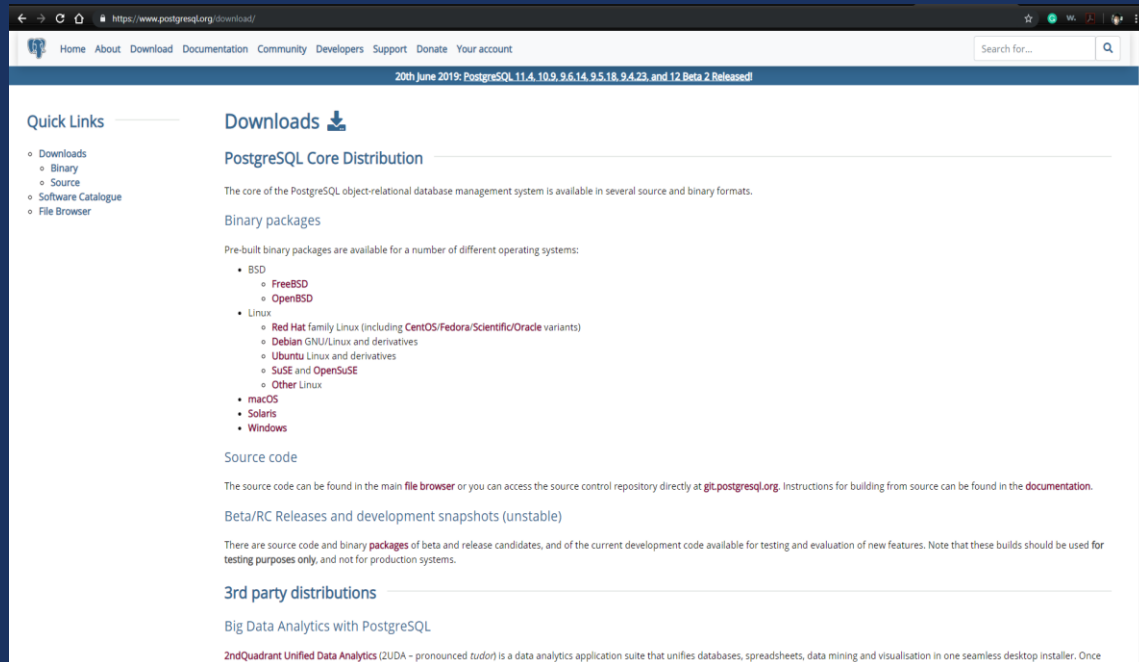
# TABLEAU

- Powerful analytical tool
- Used for business intelligence
- Connects directly to databases
- Ability to use custom SQL to obtain results
- Has built in functions to help with modeling and forecasting
  - Allows for the ability to use one dataset for the entire report





# CREATING THE DATABASE



The screenshot shows the PostgreSQL website's download section. It features a navigation bar with links like Home, About, Download, Documentation, Community, Developers, Support, Donate, and Your account. A banner at the top of the main content area reads "20th June 2019: PostgreSQL 11.4, 10.9, 9.6.14, 9.5.18, 9.4.23, and 12 Beta 2 Released!". Below this, there are sections for "Quick Links" (Downloads, Binary, Source, Software Catalogue, File Browser), "Downloads" (PostgreSQL Core Distribution, Binary packages, Source code), and "3rd party distributions" (Big Data Analytics with PostgreSQL, 2ndQuadrant Unified Data Analytics).

20th June 2019: PostgreSQL 11.4, 10.9, 9.6.14, 9.5.18, 9.4.23, and 12 Beta 2 Released!

## Quick Links

- Downloads
- Binary
- Source
- Software Catalogue
- File Browser

## Downloads

### PostgreSQL Core Distribution

The core of the PostgreSQL object-relational database management system is available in several source and binary formats.

### Binary packages

Pre-built binary packages are available for a number of different operating systems:

- BSD
  - FreeBSD
  - OpenBSD
- Linux
  - Red Hat family Linux (including CentOS/Fedora/Scientific/Oracle variants)
  - Debian GNU/Linux and derivatives
  - Ubuntu Linux and derivatives
  - SUSE and OpenSUSE
  - Other Linux
- macOS
- Solaris
- Windows

### Source code

The source code can be found in the main **file browser** or you can access the source control repository directly at [git.postgresql.org](https://git.postgresql.org). Instructions for building from source can be found in the **documentation**.

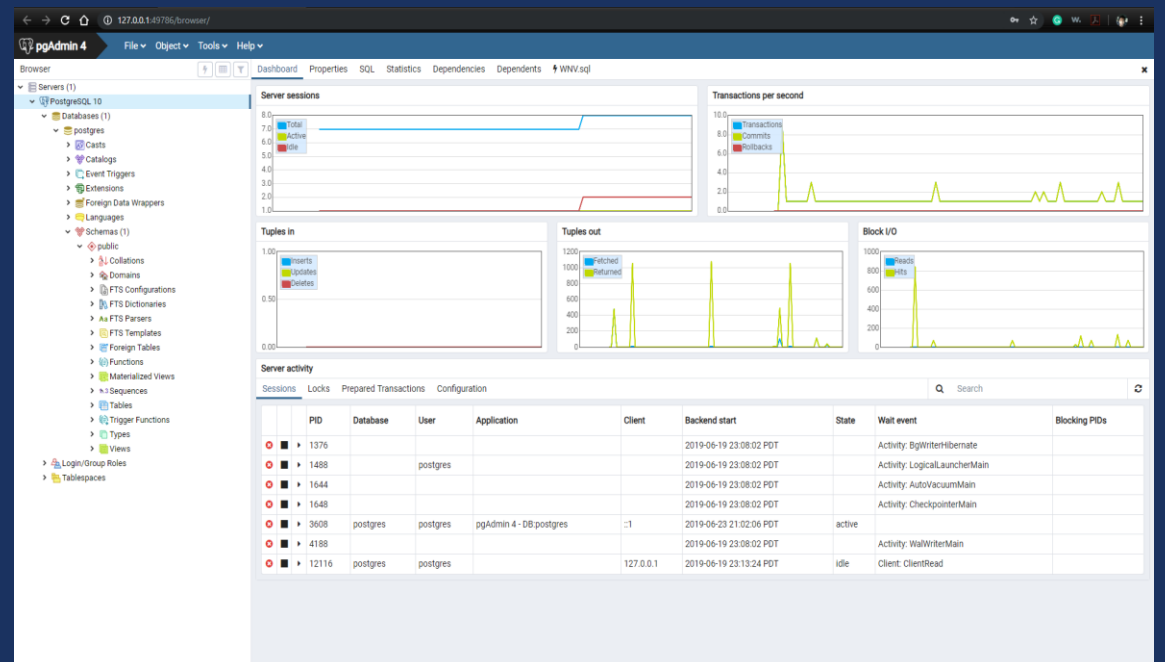
### Beta/RC Releases and development snapshots (unstable)

There are source code and binary **packages** of beta and release candidates, and of the current development code available for testing and evaluation of new features. Note that these builds should be used for **testing purposes only**, and not for production systems.

### 3rd party distributions

#### Big Data Analytics with PostgreSQL

**2ndQuadrant Unified Data Analytics** (2UDA - pronounced *tudor*) is a data analytics application suite that unifies databases, spreadsheets, data mining and visualisation in one seamless desktop installer. Once



The screenshot shows the pgAdmin 4 web interface. The left sidebar displays a tree view of the database structure, including Servers, Databases, Schemas, and Tables. The main panel shows several performance graphs: "Server sessions" (Total, Active, Idle), "Transactions per second" (Transactions, Commits, Rollbacks), "Tuples in" (Inserts, Updates, Deletes), "Tuples out" (Fetched, Returned), and "Block I/O" (Reads, Hits). Below the graphs is a "Server activity" table showing active sessions.

	PID	Database	User	Application	Client	Backend start	State	Wait event	Blocking PIDs
●	1376					2019-06-19 23:08:02 PDT		Activity: BgWriterHibernate	
●	1488		postgres			2019-06-19 23:08:02 PDT		Activity: LogicalLauncherMain	
●	1644					2019-06-19 23:08:02 PDT		Activity: AutoVacuumMain	
●	1648					2019-06-19 23:08:02 PDT		Activity: CheckpointerMain	
●	3608	postgres	postgres	pgAdmin 4 - DB postgres	:1	2019-06-23 21:02:06 PDT	active		
●	4188					2019-06-19 23:08:02 PDT		Activity: WalWriterMain	
●	12116	postgres	postgres		127.0.0.1	2019-06-19 23:13:24 PDT	idle	Client: ClientRead	

# CODE REVIEW

## ■ Info.py

- Used to store admin credentials and PostgreSQL server information as variables

## ■ Import\_Function.py

- Used to generate a create table SQL statement based on the headers and data stored in a CSV file

## ■ Import\_CSV.py

- Used to obtain all CSV files within a directory and run the functions built in Import\_Function.py to create the table and upload CSV data to its unique table. The code then moves the files to an archive directory.

```
1  #log on information
2
3  #Test log on information
4  User="postgres"
5  Password="google"
6  Host="127.0.0.1"
7  Port="5432"
8  Database="postgres"
9  |
```

INFO.PY

# IMPORT\_FUNCTION.PY

```
1 import csv, ast, psycopg2
2 from info import *
3
4
5
6 def run_test(fullpath,name):
7     f=open(fullpath, 'r')
8     reader = csv.reader(f)
9     longest, headers, type_list=[],[],[]
10
11     def dataType(val, current_type):
12         try:
13             t=ast.literal_eval(val)
14         except ValueError:
15             return 'varchar'
16         except SyntaxError:
17             return 'varchar'
18         if type(t) in [int,long,float]:
19             if (type(t) in [int, long]) and current_type not in ['float','varchar']:
20                 if (-32768 < t < 32767) and current_type not in ['int','bigint']:
21                     return 'smallint'
22                 elif (-2147483648 < t < 2147483647) and current_type not in ['bigint']:
23                     return 'int'
24             else:
25                 return 'bigint'
26             if type(t) is float and current_type not in ['varchar']:
27                 return 'decimal'
28         else:
29             return 'varchar'
30
31     for row in reader:
32         if len(headers) ==0:
33             headers = row
34             for col in row:
35                 longest.append(0)
36                 type_list.append(' ')
37         else:
38             for i in range(len(row)):
39                 if type_list[i] == 'varchar' or row[i] == 'NA':
40                     pass
41                 else:
42                     var_type = dataType(row[i],type_list[i])
43                     type_list[i] = var_type
44                     if len(row[i]) > longest[i]:
45                         longest[i] = len(row[i])
46
47     f.close()
48
49     statement = 'create table '+name+'('
50
51     for i in range(len(headers)):
52         if type_list[i] == 'varchar':
```

```
42         var_type = dataType(row[i],type_list[i])
43         type_list[i] = var_type
44         if len(row[i]) > longest[i]:
45             longest[i] = len(row[i])
46
47     f.close()
48
49     statement = 'create table '+name+'('
50
51     for i in range(len(headers)):
52         if type_list[i] == 'varchar':
53             statement = (statement + '\n{} varchar({}),'.format(headers[i].lower(), str(longest[i])))
54         else:
55             statement = (statement + '\n' + '{} {} '.format(headers[i].lower(), type_list[i]))
56
57     statement = statement[:-1]+');'
58
59     #Commented code is was used for testing purposes to drop table and recreate
60
61     #connection = psycopg2.connect(user = User, password = Password, host = Host, port = Port, database = Database)
62     #cursor = connection.cursor()
63     #cursor.execute('Drop table '+name)
64     #connection.commit()
65     #print("Table dropped PostgreSQL ")
66     #cursor.close()
67     #connection.close()
68
69     connection = psycopg2.connect(user = User, password = Password, host = Host, port = Port, database = Database)
70     cursor = connection.cursor()
71     cursor.execute(statement)
72     connection.commit()
73     print (statement)
74     print("Table created successfully in PostgreSQL ")
75     cursor.close()
76     connection.close()
77
78
79     connection = psycopg2.connect(user = User, password = Password, host = Host, port = Port, database = Database)
80     cursor = connection.cursor()
81     cursor.execute('copy PUBLIC.'+name+ ' from '+''+fullpath+''' with delimiter ',' csv header;')
82     connection.commit()
83     print("Data uploaded to PostgreSQL ")
84     cursor.close()
85     connection.close()
```

```
1 import os
2 import shutil
3 from Import_Function import *
4 path = "C:\\Users\\eltac\\Desktop\\Regis_Homework\\Test_Folder\\Test_Data"
5 Archive="C:\\Users\\eltac\\Desktop\\Regis_Homework\\Test_Folder\\Archive"
6 def files(path):
7     for file in os.listdir(path):
8         if os.path.isfile(os.path.join(path, file)):
9             yield file
10
11 for file in files(path):
12     csvfile = file
13     name = csvfile.replace(".csv", "")
14     fullpath = (path + '\\\\' + csvfile)
15     ArchivePath = (Archive + '\\\\' + csvfile)
16     print(csvfile)
17     print (name)
18     print (fullpath)
19     run_test(fullpath, name)
20     shutil.move(fullpath, ArchivePath)
```

# IMPORT\_CSV.PY

# ANALYSIS

- Analysis done to test database and code
- Looked to see if there was a connection between rainfall and confirmed West Nile cases in Las Angeles California
- West Nile virus is a common mosquito borne illness that occurs in the United States
- Previous studies have shown that rainfall effects the Culex mosquito population
- California has seen an increase number of West Nile cases

# ANALYSIS: COLLECT DATA

**HealthData.gov**

search

AboutDatasetsDevelopersFeedback

[Home](#) / [Datasets](#) / West Nile Virus Cases, 2006-present

[View published](#)[Add Feedback](#)



State of California

License

<http://opendefinition.org/licenses/odc-odbl/>

Other Access

The information on this page (the dataset metadata) is also available in these formats.

[JSON](#)[RDF](#)

via the [DKAN API](#)

## West Nile Virus Cases, 2006-present

State

This dataset contains positive cases of West Nile virus found in humans by county of residence, 2006-present. Humans usually become infected with West Nile virus by being bitten by an infected mosquito. Viruses carried in the mosquito's saliva enter the blood stream and local tissues where they infect immune cells. Most of the people who do become sick during a WNV infection develop what is referred to as "West Nile fever." A small percentage of people will develop a much more serious illness called West Nile neuroinvasive disease (WNNND). Positive cases in this dataset include both West Nile fever and West Nile neuroinvasive disease.

Source: [chhs.data.ca.gov](https://chhs.data.ca.gov)

Data and Resources

 West Nile Virus Cases, 2006-present (CSV)

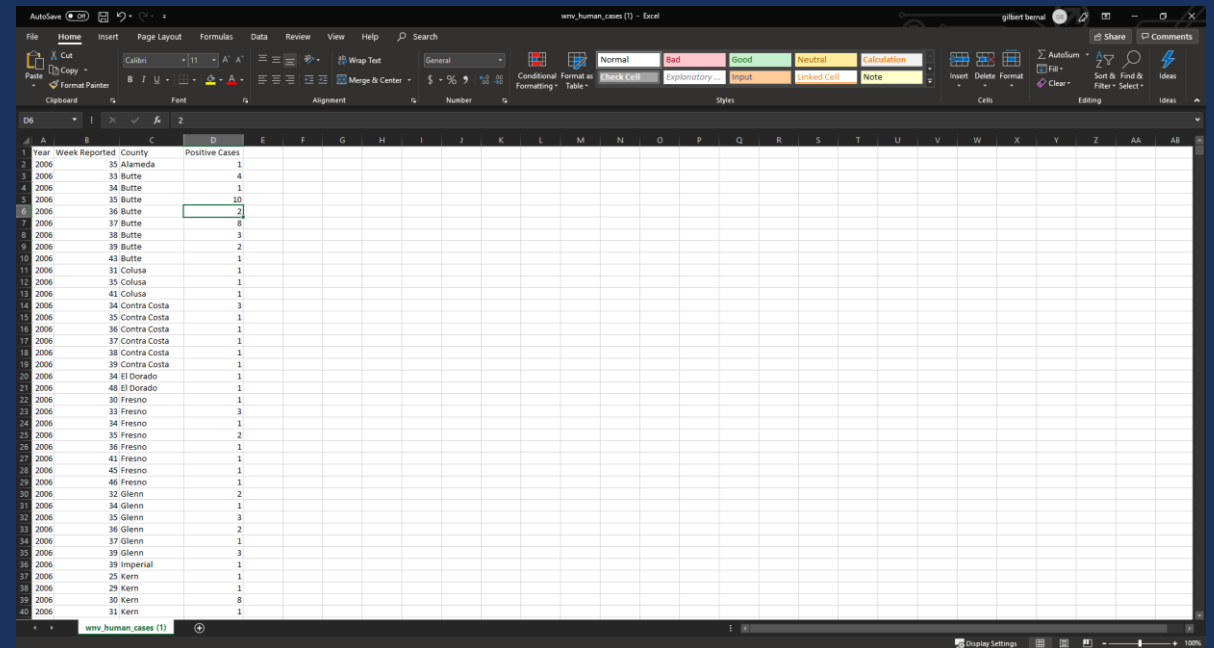
[Preview](#)[Download](#)

 Data Dictionary - West Nile Virus 2006-present

[Preview](#)[Download](#)

human casesmosquitowest nile viruswnv


Field	Value
Publisher	State of California



The screenshot shows an Excel spreadsheet titled "wnv\_human\_cases (1) - Excel". The data is organized in columns: Year, Week Reported, County, and Positive Cases. The rows list data for the year 2006, starting from week 1 and ending at week 40. The counties listed include Alameda, Butte, Colusa, Contra Costa, El Dorado, Fresno, Glenn, Imperial, Kern, and Kern. The "Positive Cases" column shows the number of cases for each entry, with values ranging from 1 to 10.

Year	Week Reported	County	Positive Cases
2006	1	Alameda	1
2006	2	Butte	4
2006	3	Butte	1
2006	4	Butte	10
2006	5	Butte	2
2006	6	Butte	8
2006	7	Butte	3
2006	8	Butte	2
2006	9	Butte	1
2006	10	Butte	1
2006	11	Colusa	1
2006	12	Colusa	1
2006	13	Colusa	1
2006	14	Contra Costa	3
2006	15	Contra Costa	1
2006	16	Contra Costa	1
2006	17	Contra Costa	1
2006	18	Contra Costa	1
2006	19	Contra Costa	1
2006	20	El Dorado	1
2006	21	El Dorado	1
2006	22	Fresno	1
2006	23	Fresno	3
2006	24	Fresno	1
2006	25	Fresno	2
2006	26	Fresno	1
2006	27	Fresno	1
2006	28	Fresno	1
2006	29	Fresno	1
2006	30	Glenn	2
2006	31	Glenn	1
2006	32	Glenn	3
2006	33	Glenn	2
2006	34	Glenn	1
2006	35	Glenn	3
2006	36	Glenn	2
2006	37	Glenn	1
2006	38	Glenn	3
2006	39	Imperial	1
2006	40	Kern	1
2006	41	Kern	1
2006	42	Kern	8
2006	43	Kern	1
2006	44	Kern	1

## ANALYSIS: COLLECT DATA



# Los Angeles Almanac™

*Since 1998*

---

**HOME   GEOGRAPHY   THE 88 CITIES   WEATHER   GOVERNMENT   MEDIA   ZIP CODES   HISTORY   COURT & COUNTY RECORDS   OTHER TOPICS**

[SEARCH](#)


**HOMEPAGE FEATURES:**

- 22 Interesting Things About Avalon & Santa Catalina Island
- Avalon Beach, 1909
- The Queen of Catalina
- "26 Miles (Santa Catalina)"
- First Surfer Hit?

[Home](#) | [All Almanac Topics](#) | [Weather](#)


## Total Seasonal Rainfall (Precipitation)

### Downtown Los Angeles (USC Campus) 1877-2018



Rainfall at the Getty Center. Photo courtesy of Monitorsystem & Prologix.com.

See also:  
[Monthly Precipitation \(Rainfall\) by Season - Downtown Los Angeles](#)  
[Record Wettest & Driest Seasons & Wettest Days - Downtown Los Angeles](#)



**Frustration-Free Email Marketing**

[Sign up for free](#)

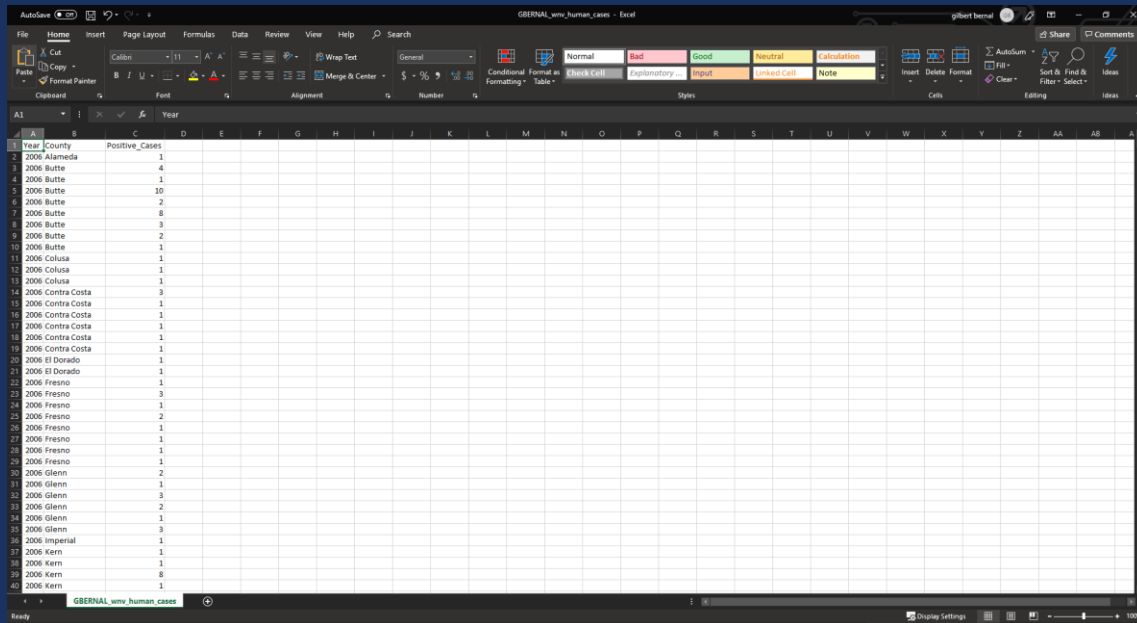
The screenshot shows the Microsoft Excel interface. The ribbon at the top includes tabs for File, Home, Insert, Page Layout, Formulas, Data, Review, View, and Help. The 'Home' tab is active, showing options for Font, Paragraph, Styles, and Editing. The formula bar at the top displays the formula  $=\text{IF}(A2 < 5, \text{Bad}, \text{IF}(A2 < 10, \text{Good}, \text{IF}(A2 < 15, \text{Neutral}, \text{IF}(A2 < 20, \text{Calculation}, \text{Note}))))$ . The worksheet contains a table with two columns: 'year' and 'rainfall'. The data is as follows:

year	rainfall
2018	4.26
2017	4.79
2016	19
2015	9.65
2014	8.52
2013	6.08
2012	9.85
2011	8.69
2010	20.2
2009	14.96
2008	9.08
2007	13.51
2006	3.21

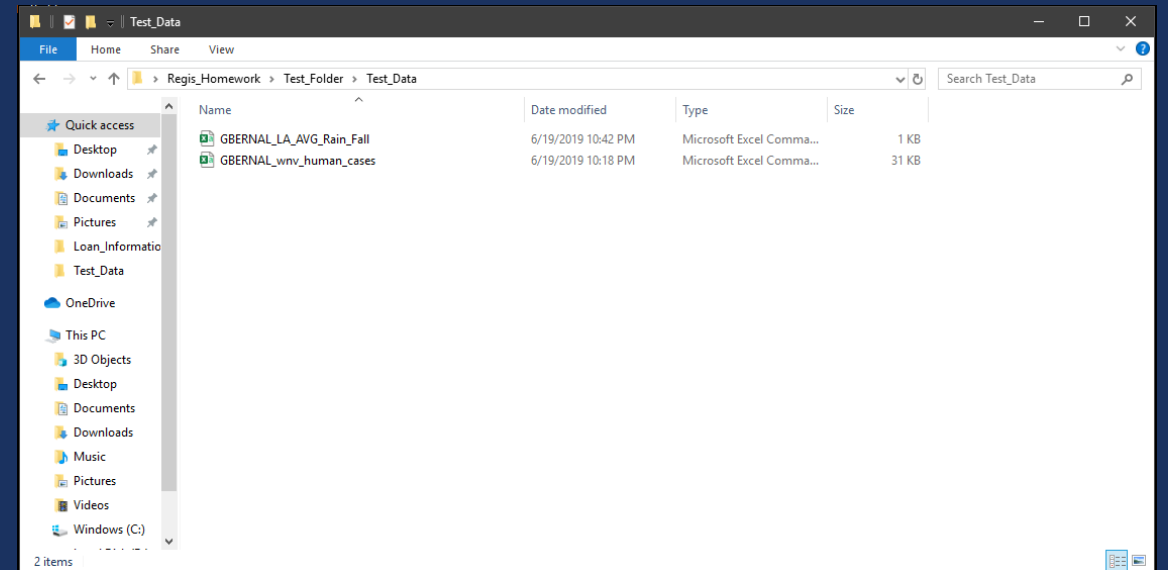
The formula bar shows the formula  $=\text{IF}(A2 < 5, \text{Bad}, \text{IF}(A2 < 10, \text{Good}, \text{IF}(A2 < 15, \text{Neutral}, \text{IF}(A2 < 20, \text{Calculation}, \text{Note}))))$ . The worksheet is titled 'Sheet1'.



# ANALYSIS: CLEAN AND RENAME FILES

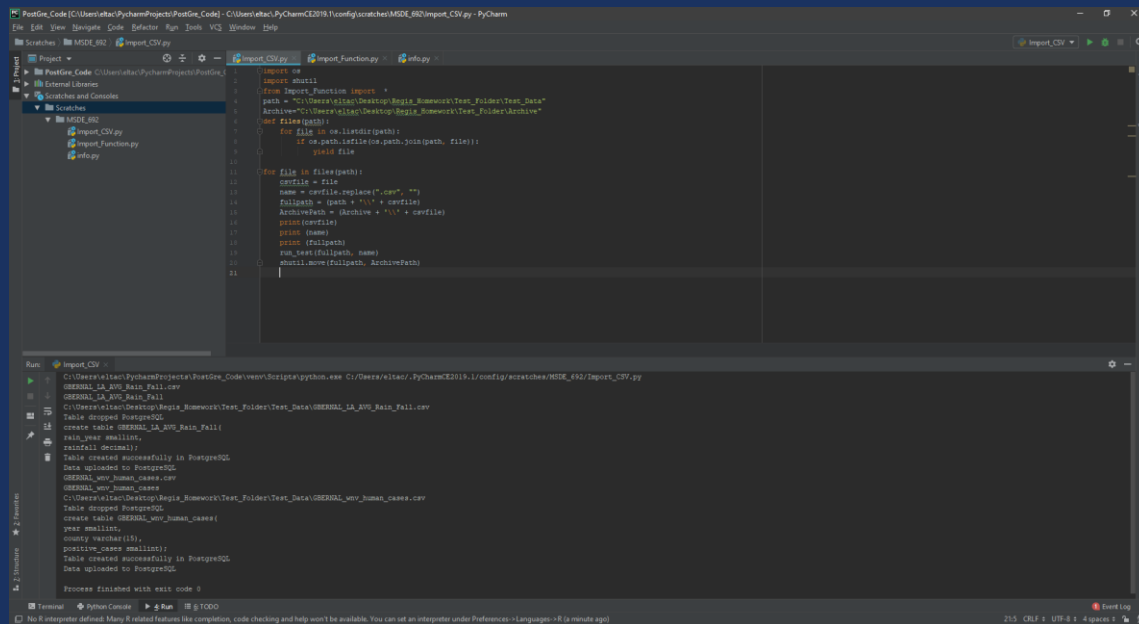


Year	County	Positive_Cases
2006	Alameda	1
2006	Butte	4
2006	Butte	1
2006	Butte	10
2006	Butte	2
2006	Butte	8
2006	Butte	3
2006	Butte	2
2006	Butte	1
2006	Colusa	1
2006	Colusa	1
2006	Colusa	1
2006	Contra Costa	3
2006	Contra Costa	1
2006	Contra Costa	1
2006	Contra Costa	1
2006	Contra Costa	1
2006	Contra Costa	1
2006	El Dorado	1
2006	Fresno	1
2006	Fresno	3
2006	Fresno	1
2006	Fresno	2
2006	Fresno	1
2006	Fresno	1
2006	Fresno	1
2006	Fresno	1
2006	Glenn	2
2006	Glenn	1
2006	Glenn	3
2006	Glenn	2
2006	Glenn	1
2006	Glenn	3
2006	Imperial	1
2006	Kern	1
2006	Kern	1
2006	Kern	8
2006	Kern	1



Name	Date modified	Type	Size
GBERNAL_LA_AVG_Rain_Fall	6/19/2019 10:42 PM	Microsoft Excel Comma...	1 KB
GBERNAL_wmv_human_cases	6/19/2019 10:18 PM	Microsoft Excel Comma...	31 KB

# ANALYSIS: RUN IMPORT\_CSV.PY



```
Python Code [C:\Users\eltac\PycharmProjects\PostGre_Code] - C:\Users\eltac\PycharmCE2019.1\config\scratches\MSDE_692\Import_CSV.py - PyCharm
In: Edit View Run Debug Run Test Run Window Help
Scratches: MSDE_692 Import_CSV.py
Project: Import_CSV.py Import_Function.py info.py
External Libraries:
Scratches and Consoles:
MSDE_692
Import_CSV.py
Import_Function.py
info.py
Run:
C:\Users\eltac\PycharmProjects\PostGre_Code\venv\Scripts\python.exe C:/Users/eltac/.PyCharmCE2019.1/config/scratches/MSDE_692/Import_CSV.py
GBERNAL_LA_AVG_Rain_Fall.csv
GBERNAL_LA_AVG_Rain_Fall
C:\Users\eltac\Desktop\Regis_Homework\Test_Folder\Test_Data\GBERNAL_LA_AVG_Rain_Fall.csv
Table dropped PostgreSQL
create table GBERNAL_LA_AVG_Rain_Fall(
rain_year smallint,
rainfall decimal);
Table created successfully in PostgreSQL
Data uploaded to PostgreSQL
GBERNAL_wmv_human_cases.csv
GBERNAL_wmv_human_cases
C:\Users\eltac\Desktop\Regis_Homework\Test_Folder\Test_Data\GBERNAL_wmv_human_cases.csv
Table dropped PostgreSQL
create table GBERNAL_wmv_human_cases(
year smallint,
county varchar(15),
positive_cases smallint);
Table created successfully in PostgreSQL
Data uploaded to PostgreSQL
Process finished with exit code 0
```

```
C:\Users\eltac\PycharmProjects\PostGre_Code\venv\Scripts\python.exe C:/Users/eltac/.PyCharmCE2019.1/config/scratches/MSDE_692/Import_CSV.py
GBERNAL_LA_AVG_Rain_Fall.csv
GBERNAL_LA_AVG_Rain_Fall
C:\Users\eltac\Desktop\Regis_Homework\Test_Folder\Test_Data\GBERNAL_LA_AVG_Rain_Fall.csv
Table dropped PostgreSQL
create table GBERNAL_LA_AVG_Rain_Fall(
rain_year smallint,
rainfall decimal);
Table created successfully in PostgreSQL
Data uploaded to PostgreSQL
GBERNAL_wmv_human_cases.csv
GBERNAL_wmv_human_cases
C:\Users\eltac\Desktop\Regis_Homework\Test_Folder\Test_Data\GBERNAL_wmv_human_cases.csv
Table dropped PostgreSQL
create table GBERNAL_wmv_human_cases(
year smallint,
county varchar(15),
positive_cases smallint);
Table created successfully in PostgreSQL
Data uploaded to PostgreSQL

Process finished with exit code 0
```

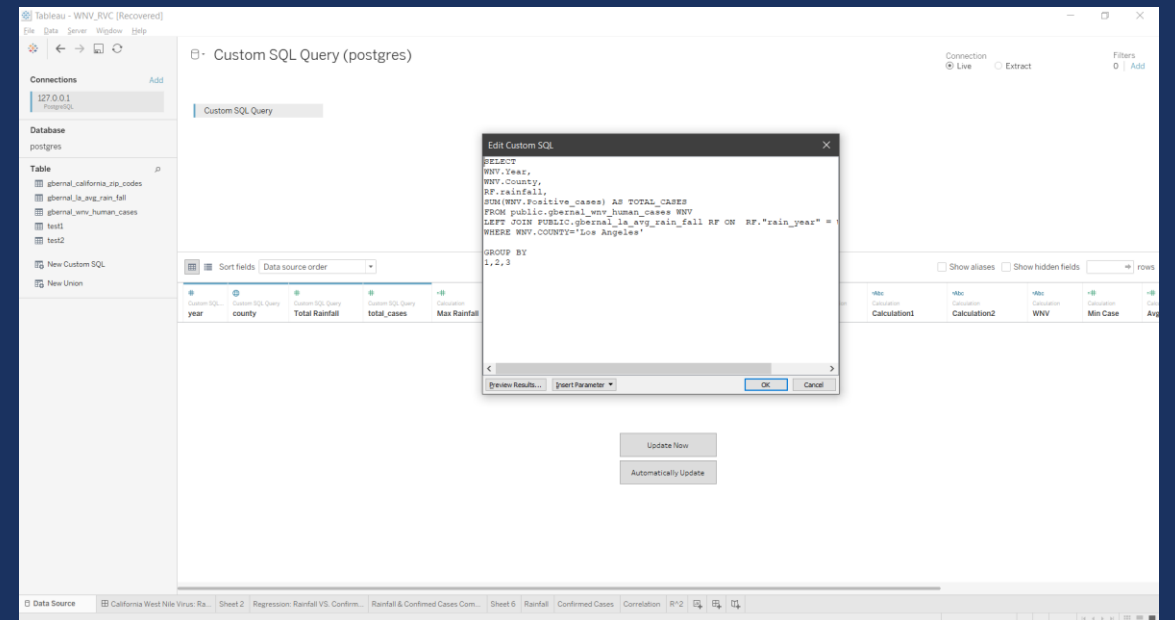
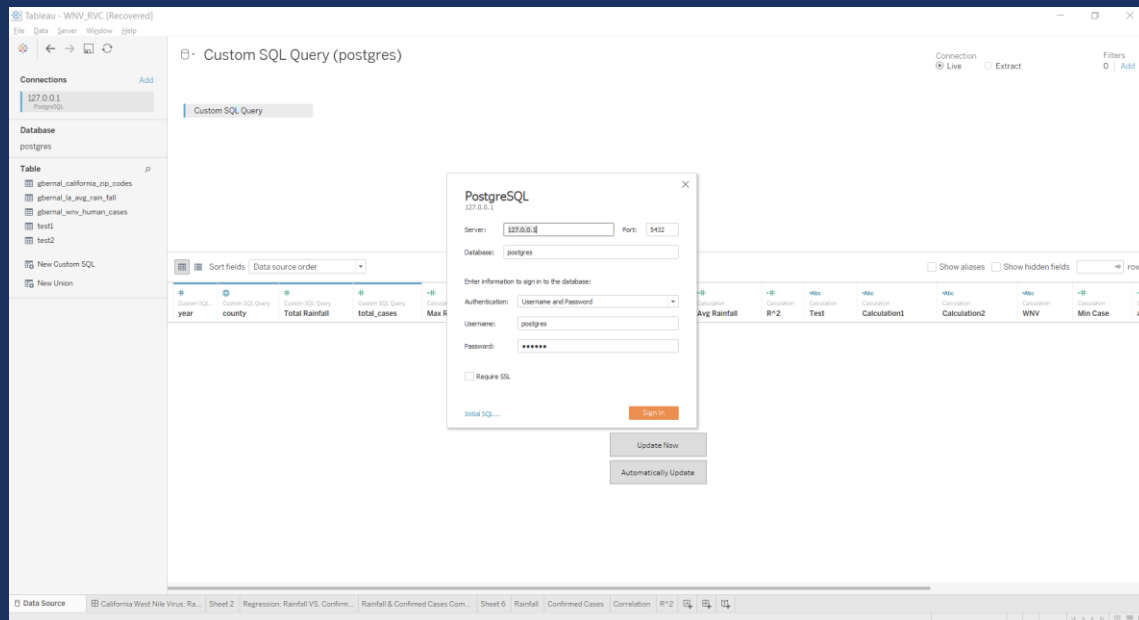
# ANALYSIS: CREATE QUERY

The screenshot displays the pgAdmin 4 web interface. On the left, the 'Servers' tree shows a PostgreSQL 10 server with a 'public' schema containing several tables. The 'Query Editor' is active, showing a SQL query that joins 'gbernal\_la\_avg\_rain\_fall' and 'gbernal\_wmv\_human\_cases' tables. The 'Data Output' tab is selected, showing a table with 13 rows of data. The table has columns: year, county, rainfall, and total\_cases. The data shows rainfall values for Los Angeles from 2006 to 2016, with total cases ranging from 4 to 277.

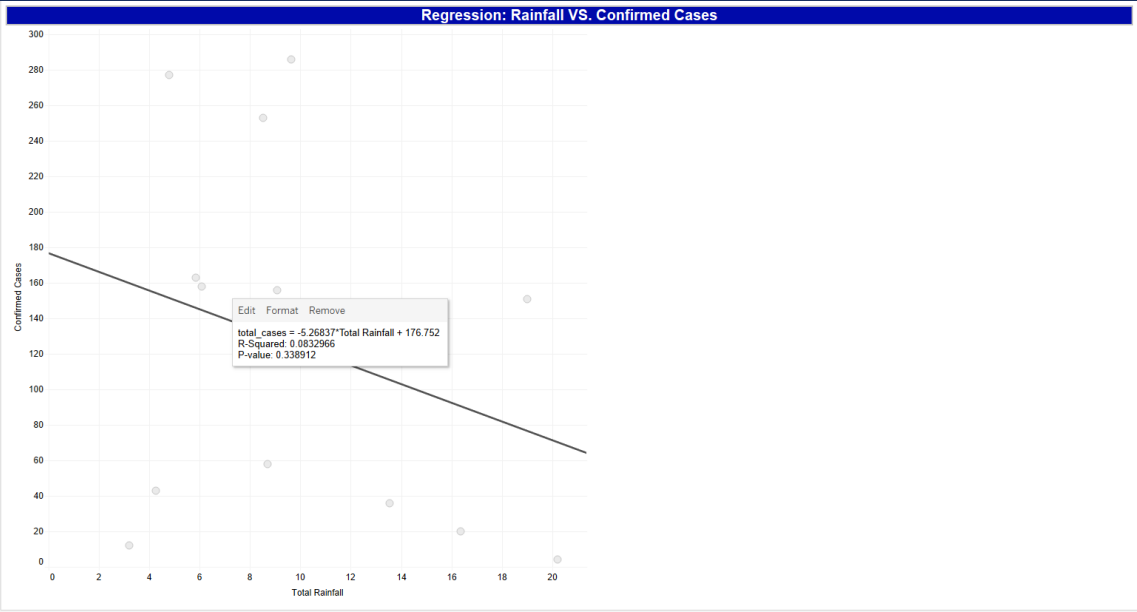
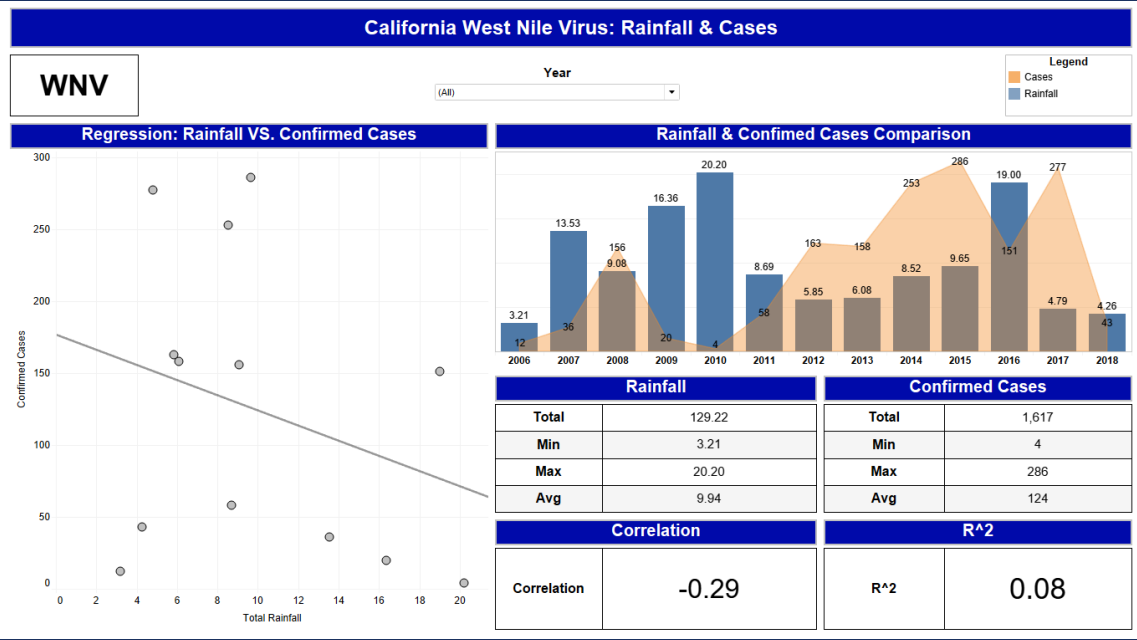
```
1 SELECT
2   WMV.Year,
3   WMV.County,
4   RF.rainfall,
5   SUM(WMV.Positive_cases) AS TOTAL_CASES
6 FROM public.gbernal_wmv_human_cases WMV
7 LEFT JOIN PUBLIC.gbernal_la_avg_rain_fall RF ON RF."rain_year" = WMV.year
8 WHERE WMV.COUNTY='Los Angeles'
9
10 GROUP BY
11   1,2,3
```

year	county	rainfall	total_cases
1	2006 Los Angeles	3.21	12
2	2012 Los Angeles	5.85	163
3	2010 Los Angeles	20.2	4
4	2014 Los Angeles	8.52	253
5	2018 Los Angeles	4.26	43
6	2013 Los Angeles	6.08	158
7	2009 Los Angeles	16.36	20
8	2017 Los Angeles	4.79	277
9	2015 Los Angeles	9.65	286
10	2011 Los Angeles	8.69	58
11	2007 Los Angeles	13.53	36
12	2008 Los Angeles	9.08	156
13	2016 Los Angeles	19	151

# ANALYSIS: CONNECT TO TABLEAU



# ANALYSIS: TABLEAU REPORT & RESULTS



# CONCLUSION

- The tools used and code created successfully worked as planned
  - PostgreSQL COPY command worked as expected
  - Code created the unique table based off the headers of the CSV file
  - Code imported the CSV data directly the unique table created
  - Database successfully connected to Tableau
  - Tableau was able to use the data to create model and report to answer analysis question

# POSSIBLE PROJECTS

- Add this project to Regis servers and add API
- Add data governance
- Update Import\_Function.py to use other data formats

# REFERENCES

- The World's Most Advanced Open Source Relational Database. (n.d.). Retrieved June 26, 2019, from <https://www.postgresql.org/>
- Total Seasonal Rainfall (Precipitation). (n.d.). Retrieved June 26, 2019, from <http://www.laalmanac.com/weather/wel3.php>
- Valdez, L. (2018, October 10). Effects of rainfall on Culex mosquito population dynamics. Retrieved June 26, 2019, from <https://arxiv.org/pdf/1703.08915.pdf>
- Welcome to Python.org. (n.d.). Retrieved June 26, 2019, from <https://www.python.org/>
- West Nile virus. (2018, December 10). Retrieved June 26, 2019, from <https://www.cdc.gov/westnile/index.html>
- West Nile Virus Cases, 2006-present. (2019, June 25). Retrieved June 26, 2019, from <https://healthdata.gov/dataset/west-nile-virus-cases-2006-present>