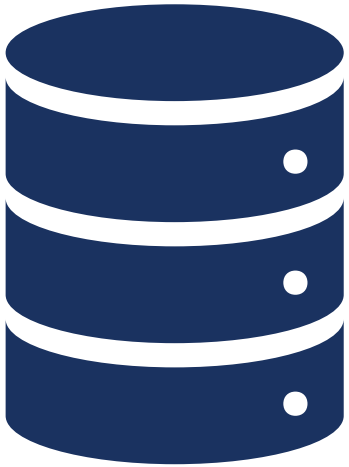


ESTABLISHMENT OF REGIS HEALTHCARE INFORMATICS DATABASE

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

Substance Abuse and Mental Health Services Administration
SAMHSA
www.samhsa.gov

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

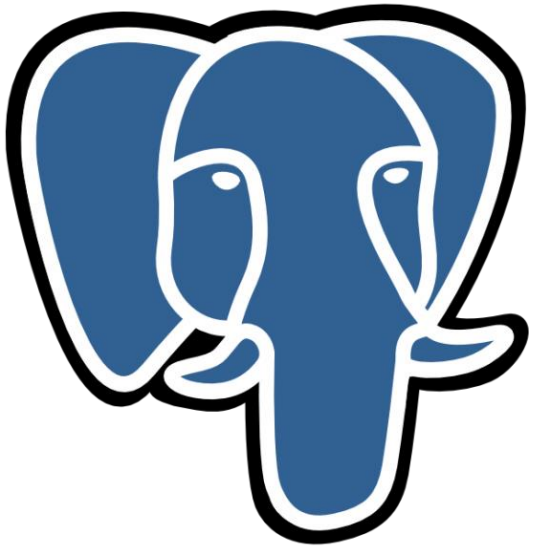
python

tableau SOFTWARE

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

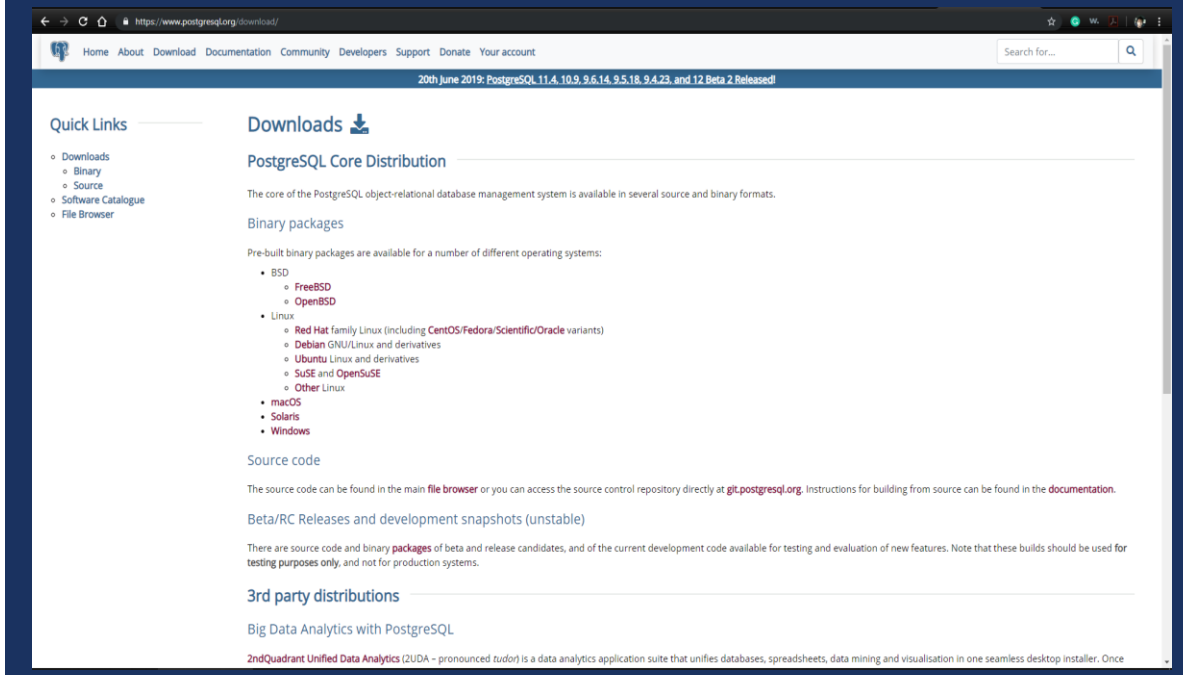
PYTHON

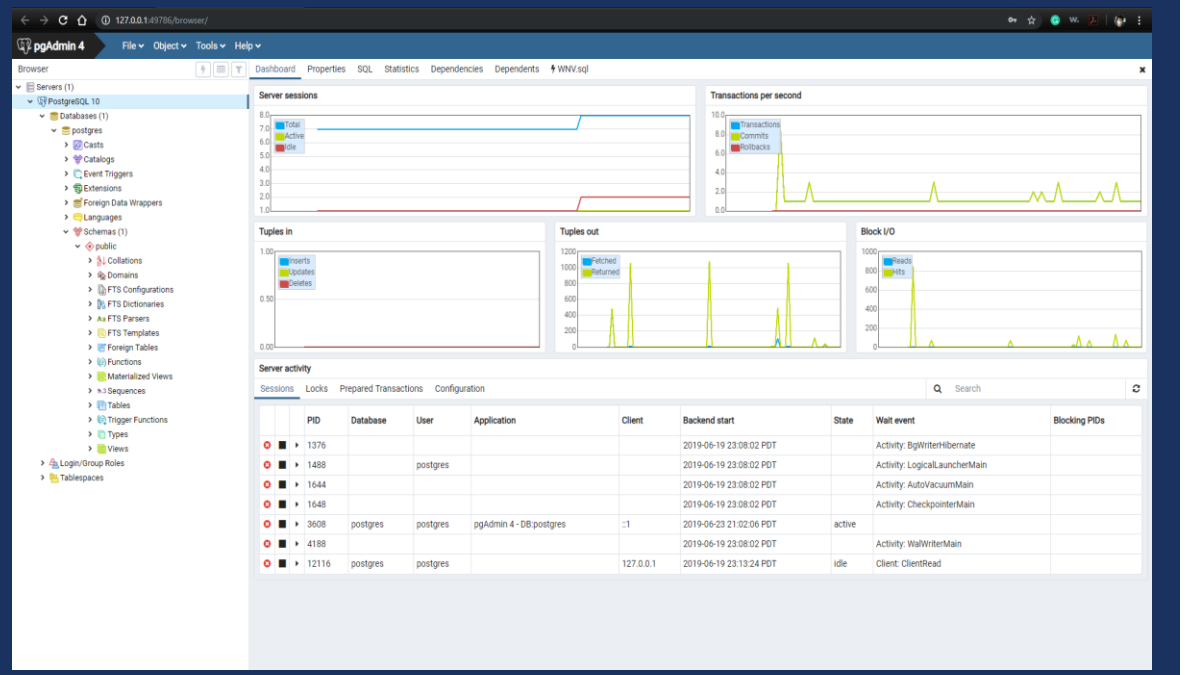
- Open source high level programming language
- Various libraries to manipulate data and files
- Works well with PostgreSQL
- Has PYCOPG2 library
  - PYCOPG2 library is the most popular PostgreSQL adaptor for Python
  - Needed in order to use the COPY command for data upload

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





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.

```python
#log on information

#Test log on information
User="postgres"
Password="google"
Host="127.0.0.1"
Port="5432"
Database="postgres"
```

INFO.PY

IMPORT_FUNCTION.PY

```python
import csv, ast, psycopg2
from info import *



def run_test(fullpath,name):
    f=open(fullpath, 'r')
    reader = csv.reader(f)
    longest, headers, type_list=[],[],[]

    def dataType(val, current_type):
        try:
            t=ast.literal_eval(val)
        except ValueError:
            return 'varchar'
        except SyntaxError:
            return 'varchar'
        if type(t) in [int,long,float]:
            if (type(t) in [int, long]) and current_type not in ['float','varchar']:
                if (-32768 < t < 32767) and current_type not in ['int','bigint']:
                    return 'smallint'
                elif (-2147483648 < t < 2147483647) and current_type not in ['bigint']:
                    return 'int'
                else:
                    return 'bigint'
            if type(t) is float and current_type not in ['varchar']:
                return 'decimal'
        else:
            return 'varchar'

    for row in reader:
        if len(headers) ==0:
            headers = row
            for col in row:
                longest.append(0)
                type_list.append(' ')
        else:
            for i in range(len(row)):
                if type_list[i] == 'varchar' or row[i] == 'NA':
                    pass
                else:
                    var_type = dataType(row[i],type_list[i])
                    type_list[i] = var_type
                if len(row[i]) > longest[i]:
                    longest[i] = len(row[i])
    f.close()

    statement = 'create table '+name+'('

    for i in range(len(headers)):
        if type_list[i] == 'varchar':
```

```python
                    var_type = dataType(row[i],type_list[i])
                    type_list[i] = var_type
                if len(row[i]) > longest[i]:
                    longest[i] = len(row[i])
    f.close()

    statement = 'create table '+name+'('

    for i in range(len(headers)):
        if type_list[i] == 'varchar':
            statement = (statement + '\n{} varchar({}),').format(headers[i].lower(), str(longest[i]))
        else:
            statement = (statement +'\n' + '{} {}' + ',').format(headers[i].lower(), type_list[i])

    statement = statement[:-1]+');'


    #Commented code is was used for testing purposes to drop table and recreate

    #connection = psycopg2.connect(user = User, password = Password, host = Host, port = Port, database = Database)
    #cursor = connection.cursor()
    #cursor.execute('Drop table '+name)
    #connection.commit()
    #print("Table dropped PostgreSQL ")
    #cursor.close()
    #connection.close()

    connection = psycopg2.connect(user = User, password = Password, host = Host, port = Port, database = Database)
    cursor = connection.cursor()
    cursor.execute(statement)
    connection.commit()
    print (statement)
    print("Table created successfully in PostgreSQL ")
    cursor.close()
    connection.close()


    connection = psycopg2.connect(user = User, password = Password, host = Host, port = Port, database = Database)
    cursor = connection.cursor()
    cursor.execute('copy PUBLIC.'+name+ ' from '+"'"+fullpath+"'"+" with delimiter ',' csv header;")
    connection.commit()
    print("Data uploaded to PostgreSQL ")
    cursor.close()
    connection.close()
```

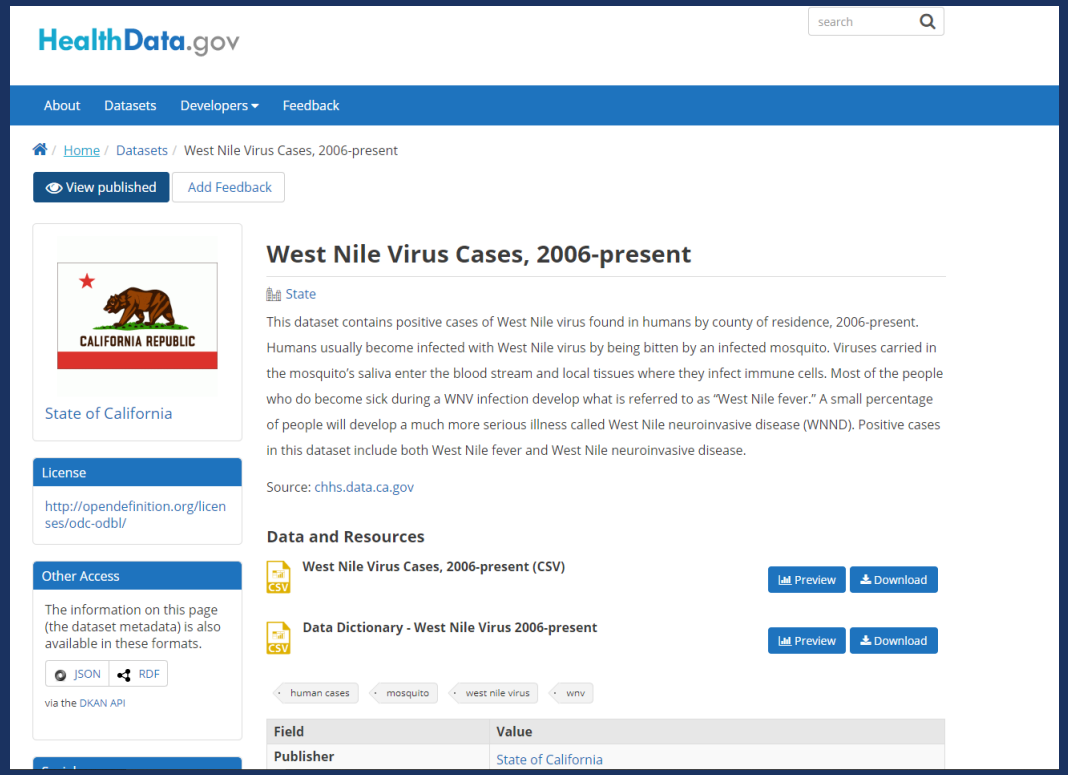
```python
import os
import shutil
from Import_Function import  *
path = "C:\Users\eltac\Desktop\Regis_Homework\Test_Folder\Test_Data"
Archive="C:\Users\eltac\Desktop\Regis_Homework\Test_Folder\Archive"
def files(path):
    for file in os.listdir(path):
        if os.path.isfile(os.path.join(path, file)):
            yield file

for file in files(path):
    csvfile = file
    name = csvfile.replace(".csv", "")
    fullpath = (path + '\\' + csvfile)
    ArchivePath = (Archive + '\\' + csvfile)
    print(csvfile)
    print (name)
    print (fullpath)
    run_test(fullpath, name)
    shutil.move(fullpath, ArchivePath)
```

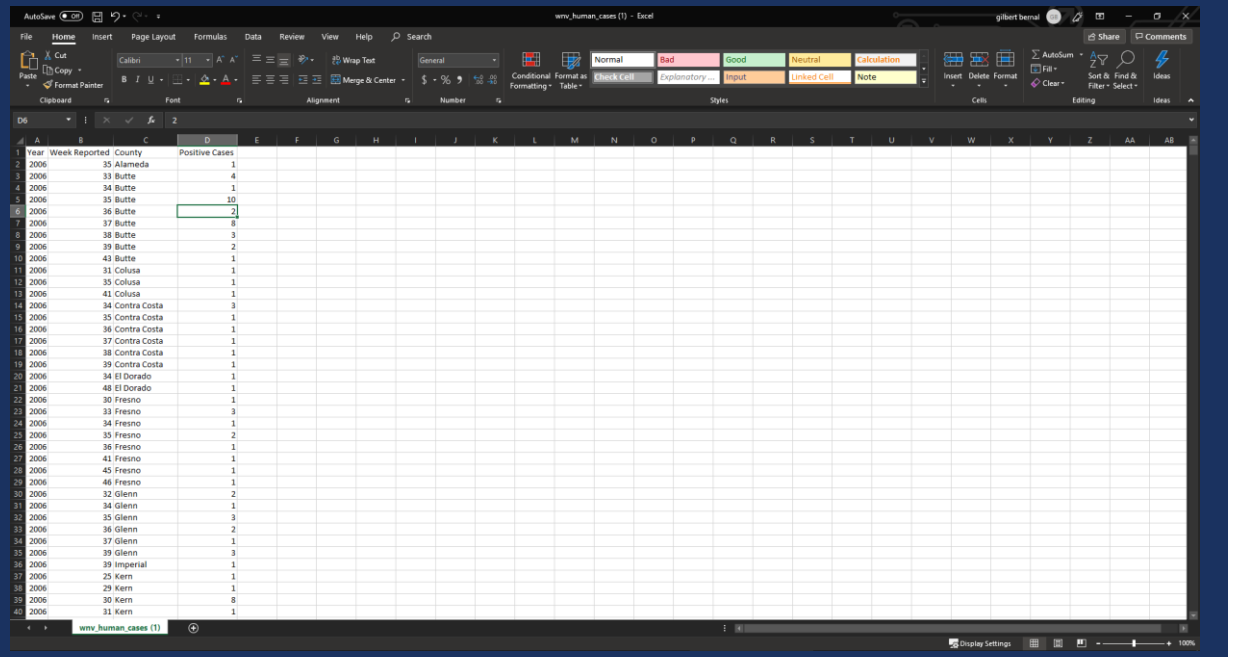
IMPORT_CSV.PY

ANALYSIS

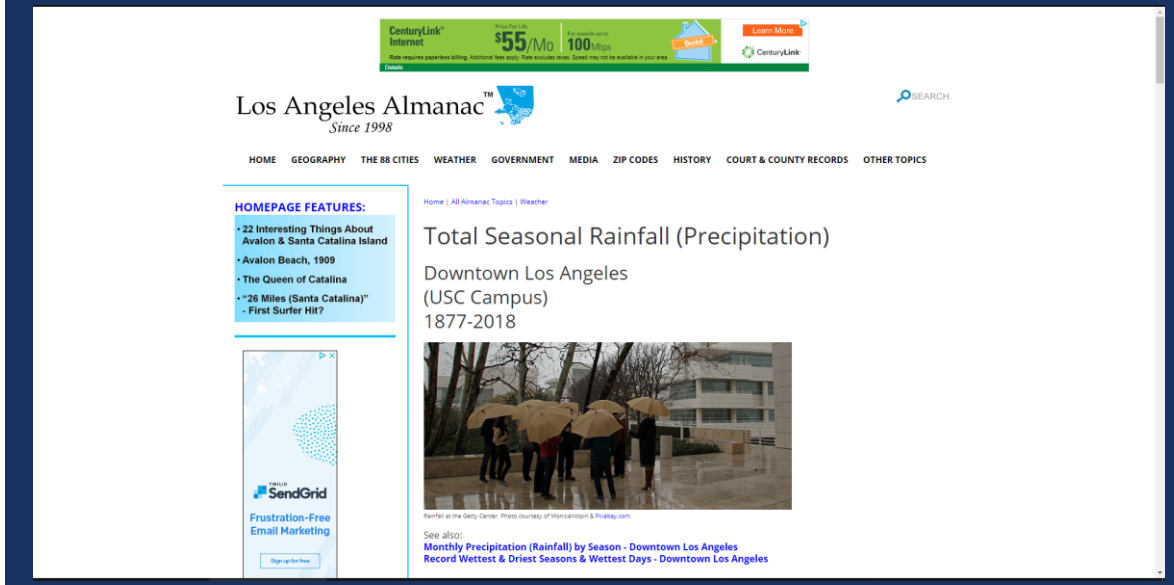
- Analysis done to test database and code
- Looked to see if there was a connection between rainfall and confirmed West Nile cases in Las Angeles California
- West Nile virus is a common mosquito borne illness that occurs in the United States
- Previous studies have shown that rainfall effects the Culex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

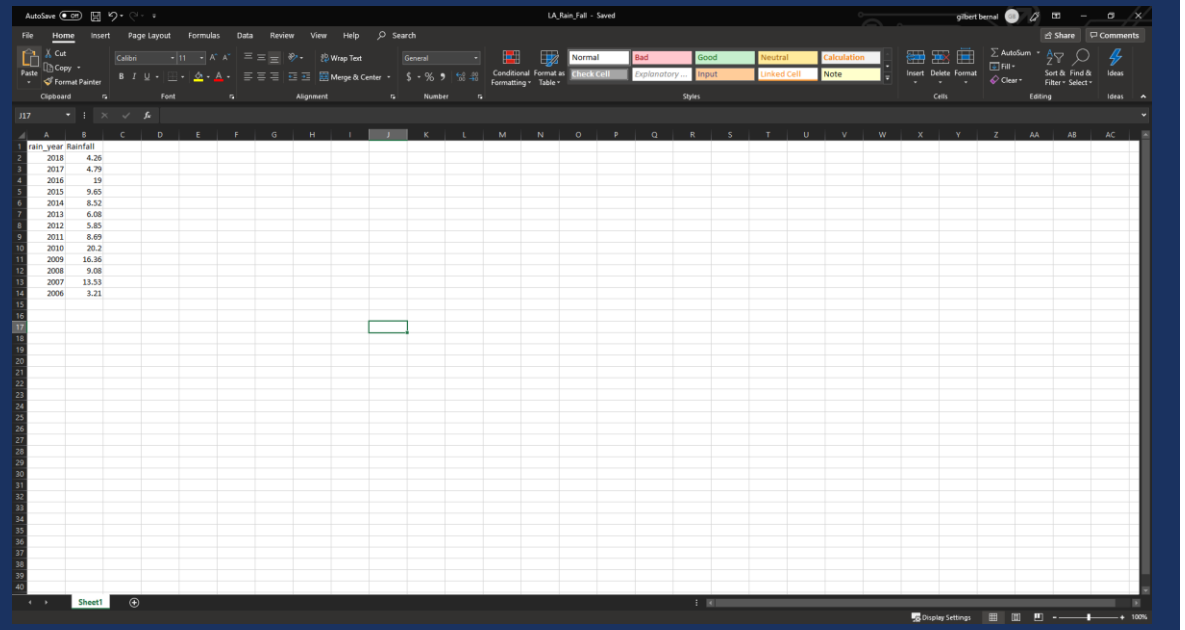
ANALYSIS: COLLECT DATA



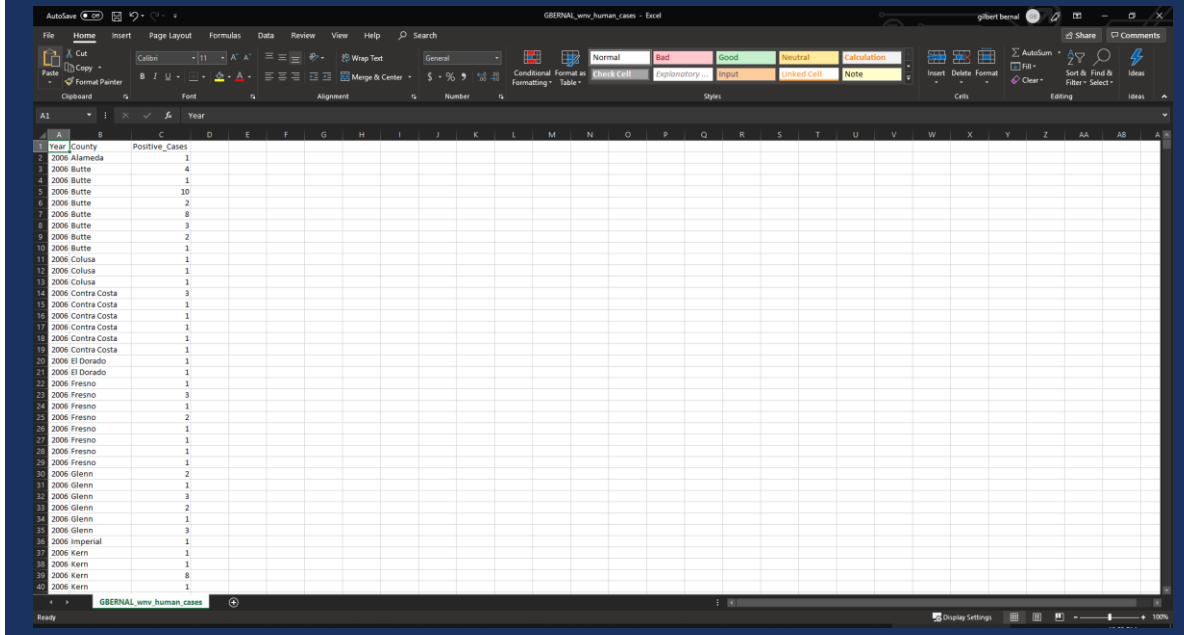


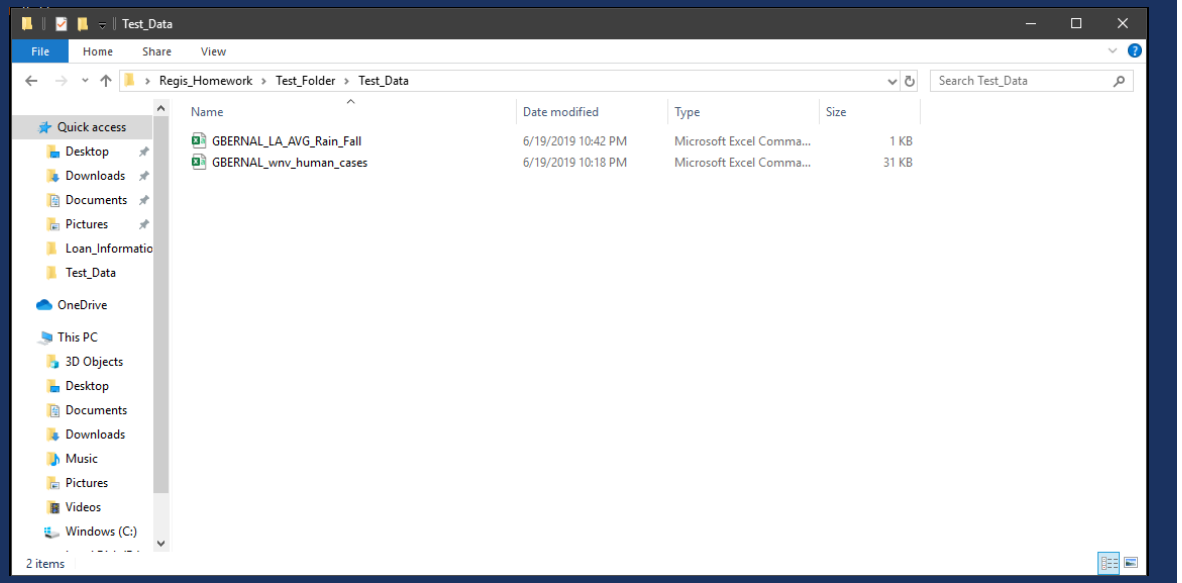
ANALYSIS: COLLECT DATA



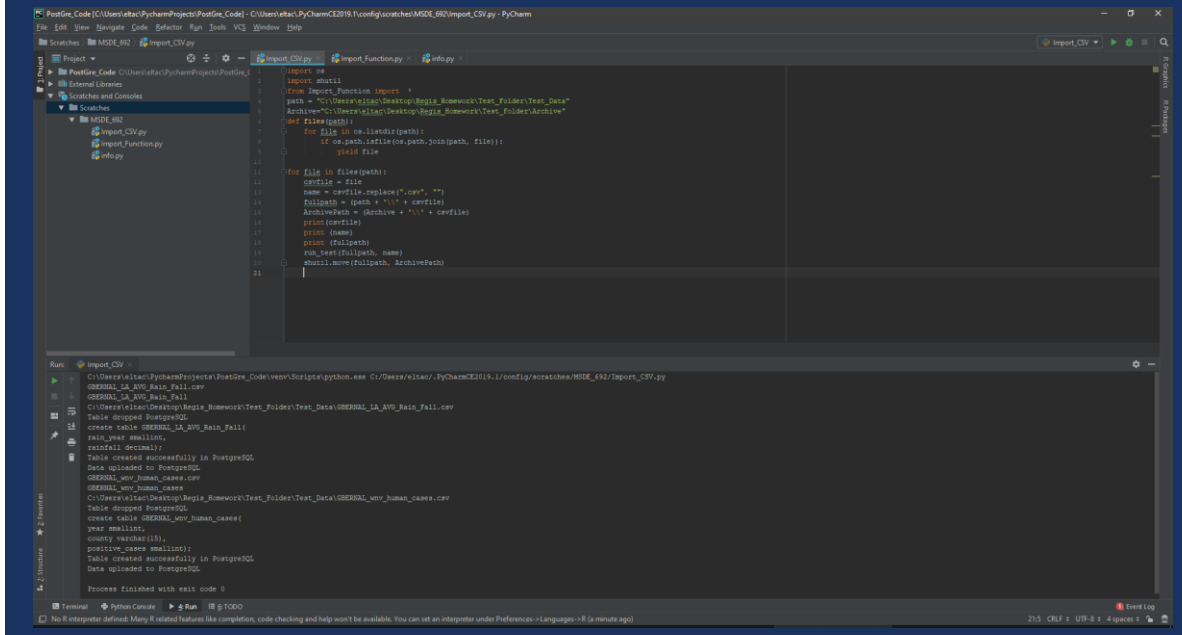


ANALYSIS: CLEAN AND RENAME FILES



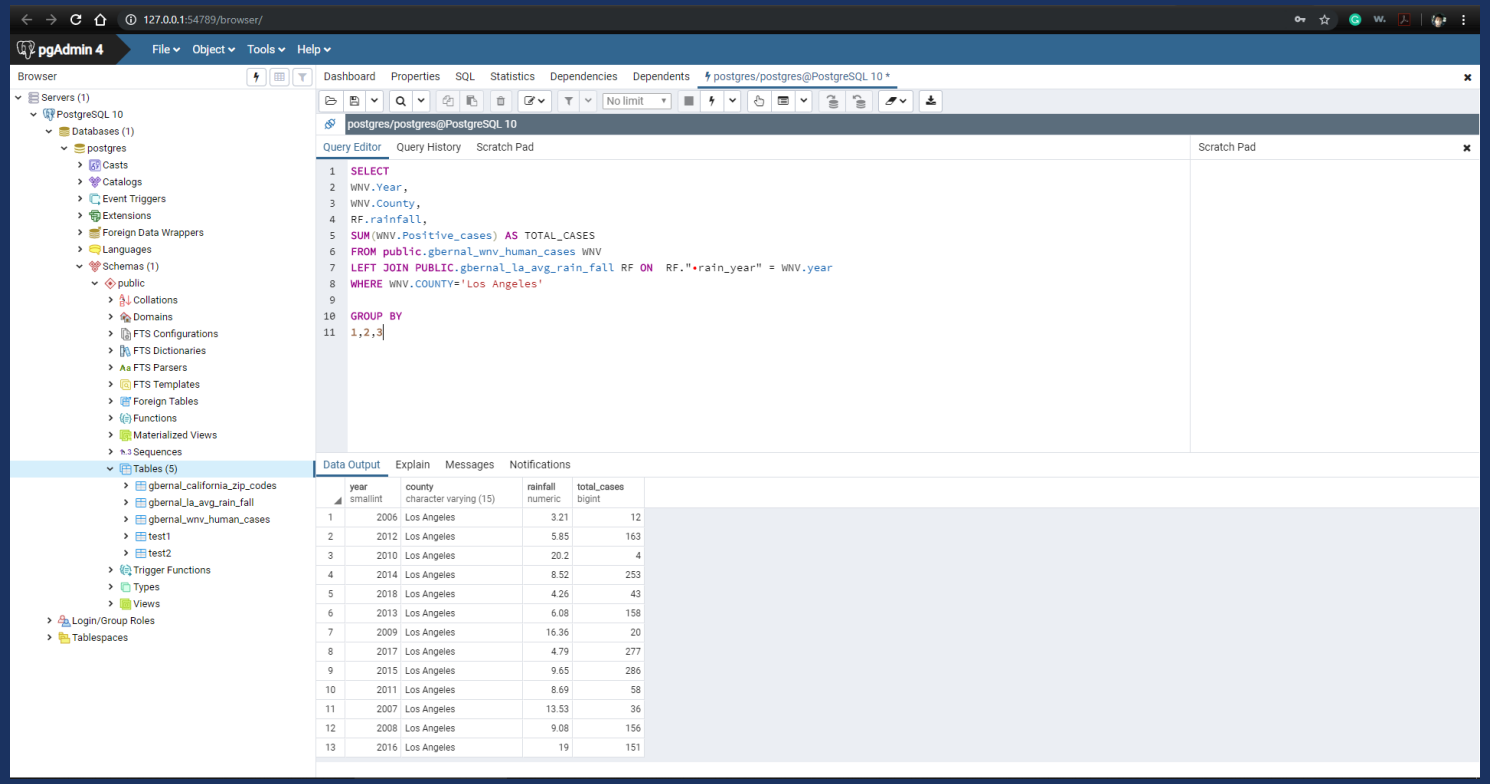


ANALYSIS: RUN IMPORT_CSV.PY

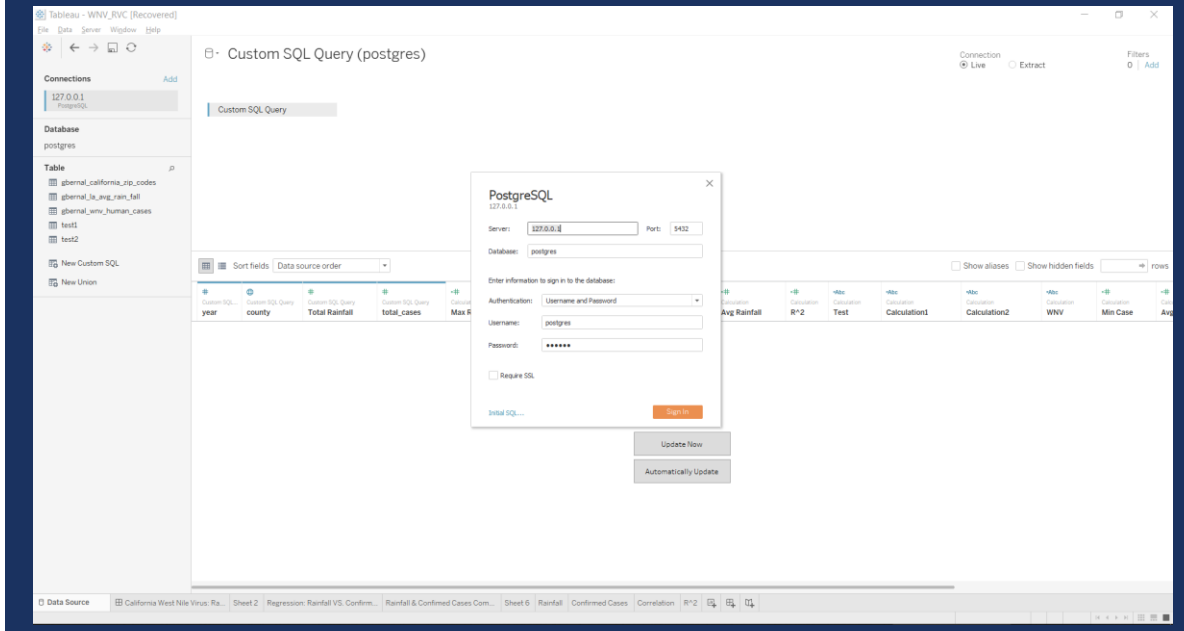


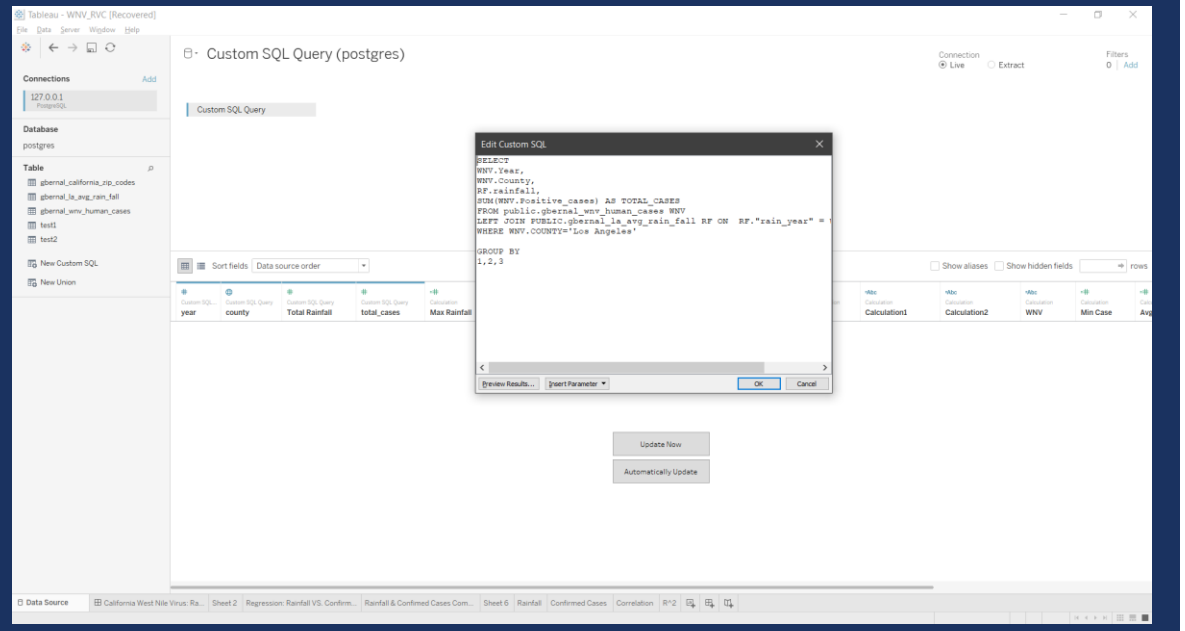

```
C:\Users\eltac\PycharmProjects\PostGre_Code\venv\Scripts\python.exe C:/Users/eltac/.PyCharmCE2019.1/config/scratches/MSDE_692/Import_CSV.py
GBERNAL_LA_AVG_Rain_Fall.csv
GBERNAL_LA_AVG_Rain_Fall
C:\Users\eltac\Desktop\Regis_Homework\Test_Folder\Test_Data\GBERNAL_LA_AVG_Rain_Fall.csv
Table dropped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

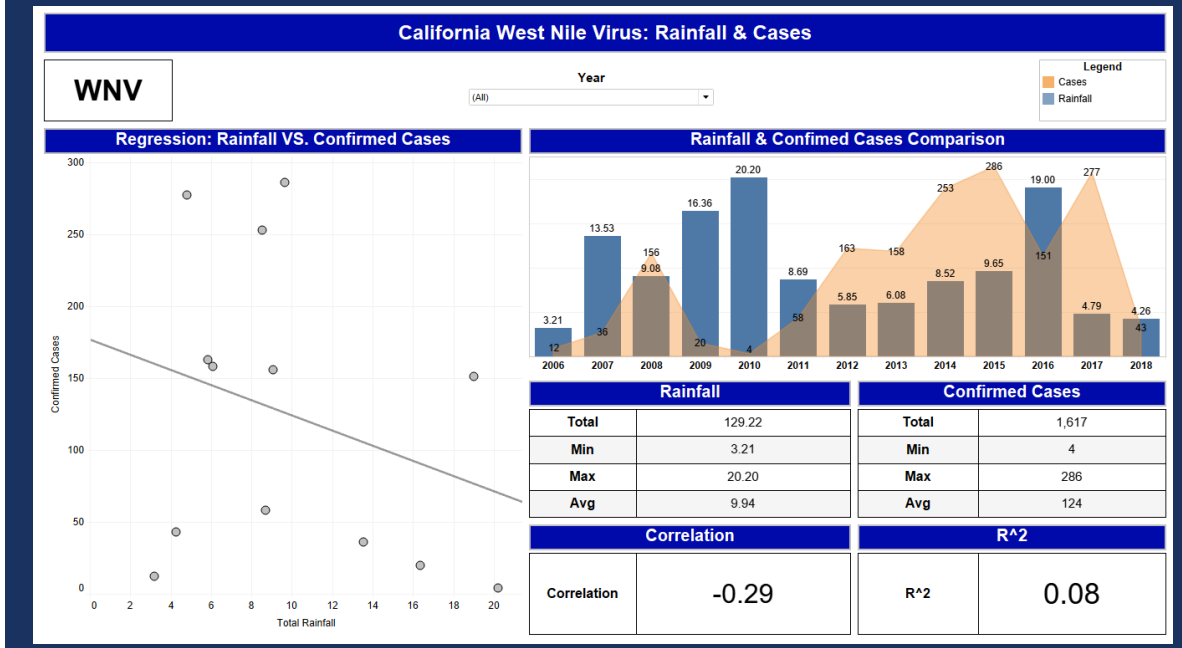


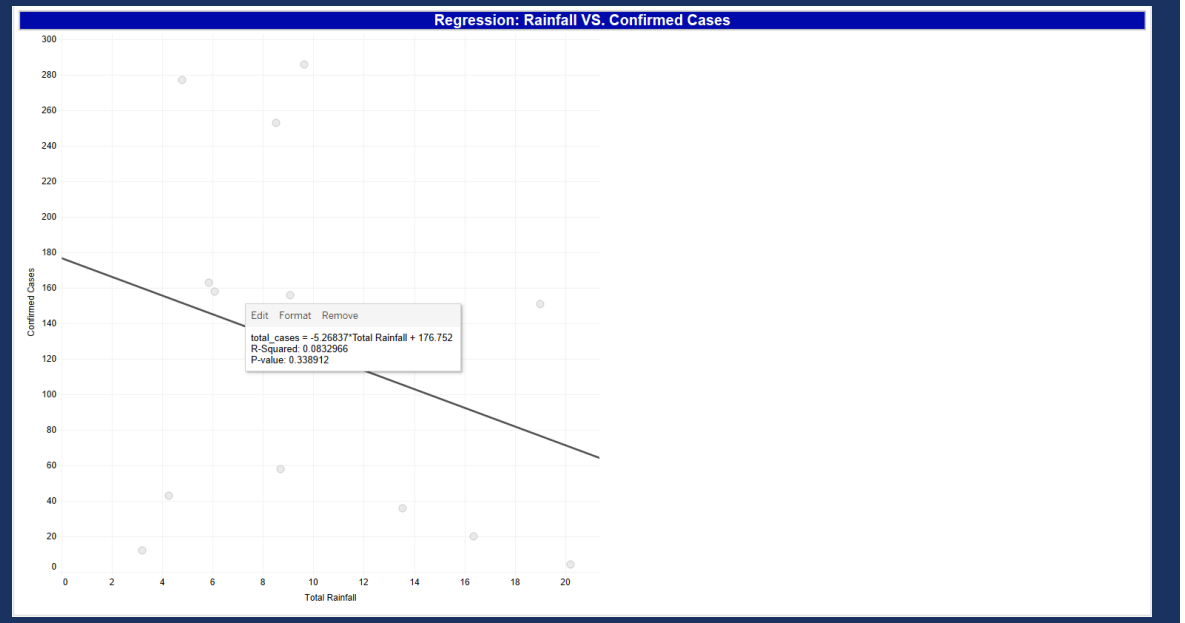
ANALYSIS: CONNECT TO TABLEAU





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 repot 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/we13.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