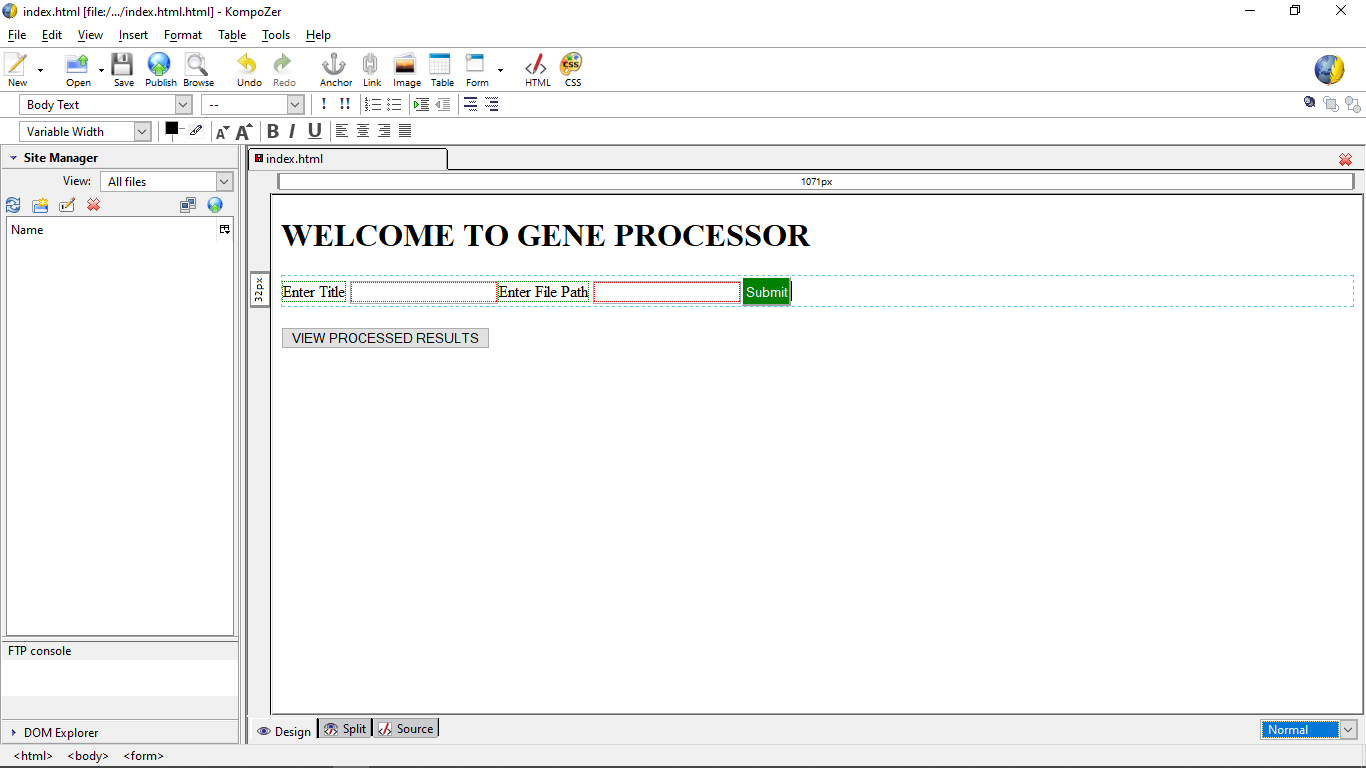
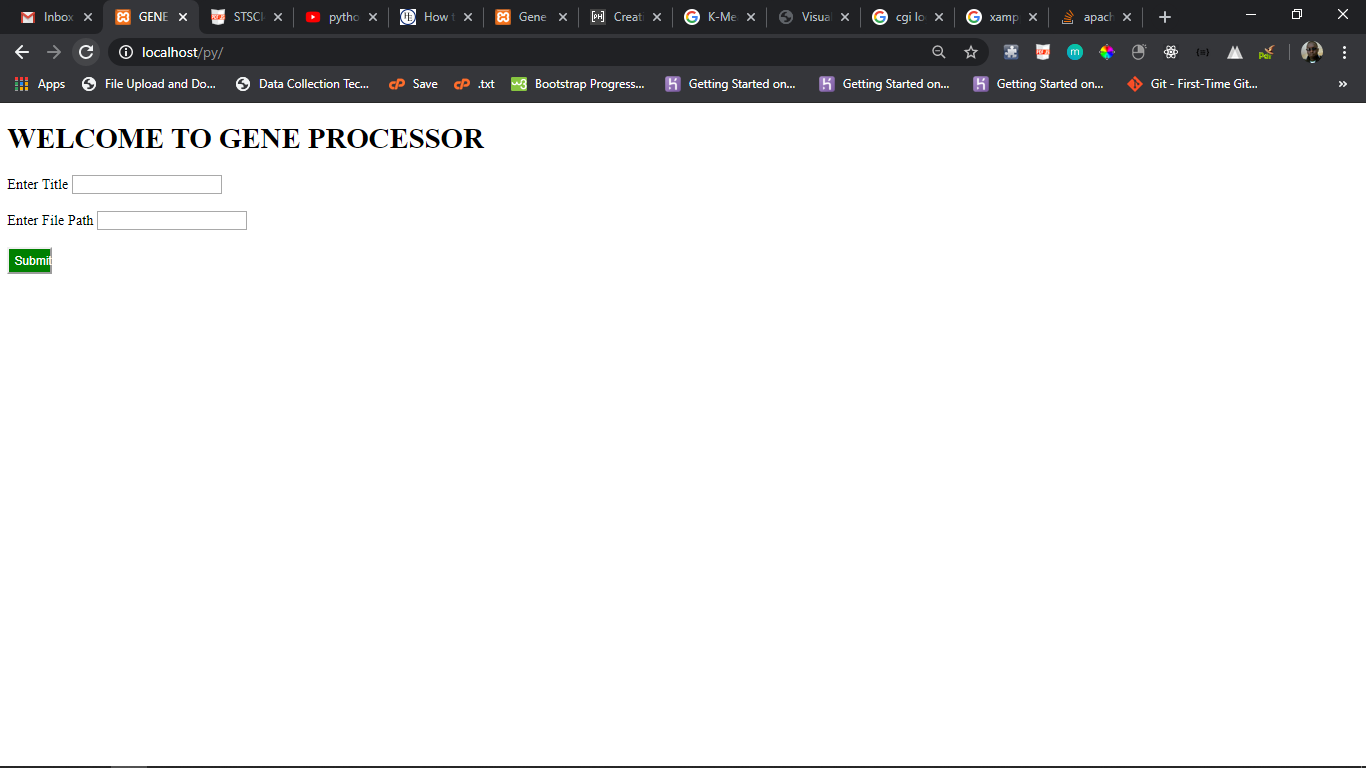
**PROJECT REPORT**

**KOMPOZER WEB PAGE WITH FORM FOR INFORMATION ENTRY I.E. FILE NAME AND FILE PATH**





**CODE 1: KOMPOZER HTML**

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">

<html><head>

  <meta content="text/html; charset=ISO-8859-1" http-equiv="content-type">

  <title>GENE PROCESSOR</title>

  <style type="text/css"></style>

</head><body>

<h1>WELCOME TO GENE PROCESSOR</h1>

<form enctype="multipart/form-data" method="POST" action="process.py" name="gene">

<label>Enter Title</label>

<input name="title"><br>

<br>

<label>Enter File Path</label>

<input name="path"><br><br>

  <button style="background-color: green; color: white; height: 30px; width: 50px;" value="Submit" name="submit">Submit</button><br>

</form>

<br>

</body>

</html>

Code 2: Oracle DB Connection

#! C:\Users\mogak\AppData\Local\Programs\Python\Python38\python.exe

#this code created the beeGenes table but is ommitted from the procee.py

print("Content-Type: text/html\n")

import cx\_Oracle

#may vary from user to user depending on installation

#create beegenes table

#please note credentials might be different for you and must be adjusted to run

conn = cx\_Oracle.connect('c##username/password@localhost:1521/orcl')

cursor = conn.cursor()

**CODE3: CREATING TABLE**

#! C:\Users\mogak\AppData\Local\Programs\Python\Python38\python.exe

#this code created the beeGenes table but is ommitted from the procee.py

print("Content-Type: text/html\n")

import cx\_Oracle

#may vary from user to user depending on installation

#create beegenes table

#please note credentials might be different for you and must be adjusted to run

conn = cx\_Oracle.connect('c##username/password@localhost:1521/orcl')

cursor = conn.cursor()

#create beegenes table

cursor.execute("CREATE TABLE beeGenes(\

    gi\_id VARCHAR (150),\

    nucleotide\_seq VARCHAR(250),\

    A NUMBER,\

    C NUMBER,\

    G NUMBER,\

    T NUMBER,\

    GC NUMBER\

        )")

if cursor:

    print("table created")

    #data = [1,6,4,5,6,7,8]

        #for row in all\_data:

    #cursor.execute("""

        #insert into beeGenes (gi\_id, nucleotide\_seq, A, C, G, T, GC)

        #values (:1, :2, :3, :4, :5, :6, :7)""", data)

    #conn.commit()

else:

    print("error creating table")

**CODE4: PROCESS.CGI**

This script takes the forma data from above and reads the text file and processes the genes, their ids and frequencies as required.

The main function takes in the form input whereas the process Input function processes the file from reading to data structuring and inserts to Oracle database

#! C:\Users\mogak\AppData\Local\Programs\Python\Python38\python.exe

print("Content-Type: text/html\n")

# Import modules for CGI handling

import cgi, cgitb

import re

#import Counter

import cx\_Oracle

from collections import defaultdict

#may vary from user to user depending on installation

#please note credentials might be different for you and must be adjusted to run

conn = cx\_Oracle.connect('c##username/password@localhost:1521/orcl')

cursor = conn.cursor()

#define main function to get input data from the form

def main():

    # instance of FieldStorage function for the form

    form = cgi.FieldStorage()

    # Get from form, title and path

    global title

    global path

    title = form.getvalue('title')

    path  = form.getvalue('path')

    #print(title)

    #print(path)

#processInput function to read, write the path file and insert into ORACLE

def processInput(file\_path, c):

    #read file to read and write

    file = open(file\_path, 'rt')

    #assign the read data to a variable

    file\_data = file.read()

    #print(file\_data)

    #text to search and append to

    text\_to\_search = 'mRNA'

    #text to append after search matches

    text\_to\_append = '\_\*\*gene\_seq\_starts\_here\*\*\_mRNA'

    #searching and appending

    file\_data = file\_data.replace(text\_to\_search, text\_to\_append)

    #closing file after operation

    file.close()

    #opening file in write mode

    file = open(file\_path, 'wt')

    #replacing the changes to the original file...this program should run once since not all

    #conditions are taken care of,,,refreshing will break the logic

    file.write(file\_data)

    #if you need to run it twice, replace with the original source text

    #closing modified file

    file.close()

    #Reading the contents of the new file to loop for occurences of

    #newfile = open(file\_path, 'r')

    #new\_data = newfile.read()

    #gi id and the gene sequence

    #extracting gene ID and sequence to ready them for db entry

    gi\_ids = []

    gene\_sequence = []

    #get gene ids----

    with open(file\_path, 'r') as infile:

        for line in infile:

            #if gi is found

            if "gi" in line:

                if line.strip().split('|')[1].strip(" ") not in gi\_ids:

                    gi\_ids.append(line.strip().split('|')[1].strip(" "))

    #loop for each line of gene matching the main henes

    with open(file\_path, 'r') as infile:

        search\_gene = ['A','C','G','T']

        for search\_gene in infile:

            if search\_gene.isupper():

                gene\_sequence.append(search\_gene.rstrip('\n'))

    #genes and ids as they appear #might be used to structure all

    #------------  i=id, j = gene

    #data before insertion have it mind

    genes = [i + j for i, j in zip(gi\_ids, gene\_sequence)]

    #------------

    #print(gi\_ids)

    #print(gene\_sequence)

    #print(genes)

    for gene in genes:

        #print(gene)

        #genes\_str = ''.join(map(str, gene))

        formarted\_gene = re.match(r'([0-9]+)([A-Z]+)', gene, re.I)

        #if formarted\_gene:#genes formated

        each\_gene = formarted\_gene.groups()#each gene has been arrived at

        #for nuc in each\_gene:

        gi\_id = each\_gene[0]

        gene\_seq = each\_gene[1]

        substringA = 'A'#A

        substringC = 'C'#C

        substringG = 'G'#G

        substringT = 'T'#T

        countA = gene\_seq.count(substringA)#FREQUENCY

        countC = gene\_seq.count(substringC)#FREQUENCY

        countG = gene\_seq.count(substringG)#FREQUENCY

        countT = gene\_seq.count(substringT)#FREQUENCY

        freq\_GC = countG + countC#FREQUENCY

        all\_data =[gi\_id,gene\_seq,countA,countC,countG,countT,freq\_GC]

        #print(all\_data)

        #data = [1,6,4,5,6,7,8]

        #for row in all\_data:

        #array size binding

        cursor.bindarraysize = 1000

        #execute entries

        cursor.execute("""

            insert into beeGenes (gi\_id, nucleotide\_seq, A, C, G, T, GC)

            values (:1, :2, :3, :4, :5, :6, :7)""", all\_data)

        conn.commit()

#def fileToStr():

#displayong the success message from the meesage template

def makePage():

    message = """<html>

    <head>

    <title>Gene Processor</title>

    </head>

    <body>

    <h1>GENE PROCESSOR</h1><br/>

    <h4 style="color:green">Your gene file has been processed successfully</h4>

    <h4 style="color:green">Your data is now available in your database</h4>

    <button><a href="fetch.py">Click to See Some Results</a></button>

    </body>

    </html>"""

    print(message)

main()

processInput(path, conn)

makePage()

#db(cursor,dba)

4)Oracle create table code

#! C:\Users\mogak\AppData\Local\Programs\Python\Python38\python.exe

#this code created the beeGenes table but is ommitted from the procee.py

print("Content-Type: text/html\n")

import cx\_Oracle

#may vary from user to user depending on installation

#create beegenes table

#please note credentials might be different for you and must be adjusted to run

conn = cx\_Oracle.connect('c##username/password@localhost:1521/orcl')

cursor = conn.cursor()

#create beegenes table

cursor.execute("CREATE TABLE beeGenes(\

    gi\_id VARCHAR (150),\

    nucleotide\_seq VARCHAR(250),\

    A NUMBER,\

    C NUMBER,\

    G NUMBER,\

    T NUMBER,\

    GC NUMBER\

        )")

if cursor:

    print("table created")

    #data = [1,6,4,5,6,7,8]

        #for row in all\_data:

    #cursor.execute("""

        #insert into beeGenes (gi\_id, nucleotide\_seq, A, C, G, T, GC)

        #values (:1, :2, :3, :4, :5, :6, :7)""", data)

    #conn.commit()

else:

    print("error creating table")

5) Oracle database connection

#! C:\Users\mogak\AppData\Local\Programs\Python\Python38\python.exe

#this code created the beeGenes table but is ommitted from the procee.py

print("Content-Type: text/html\n")

import cx\_Oracle

#may vary from user to user depending on installation

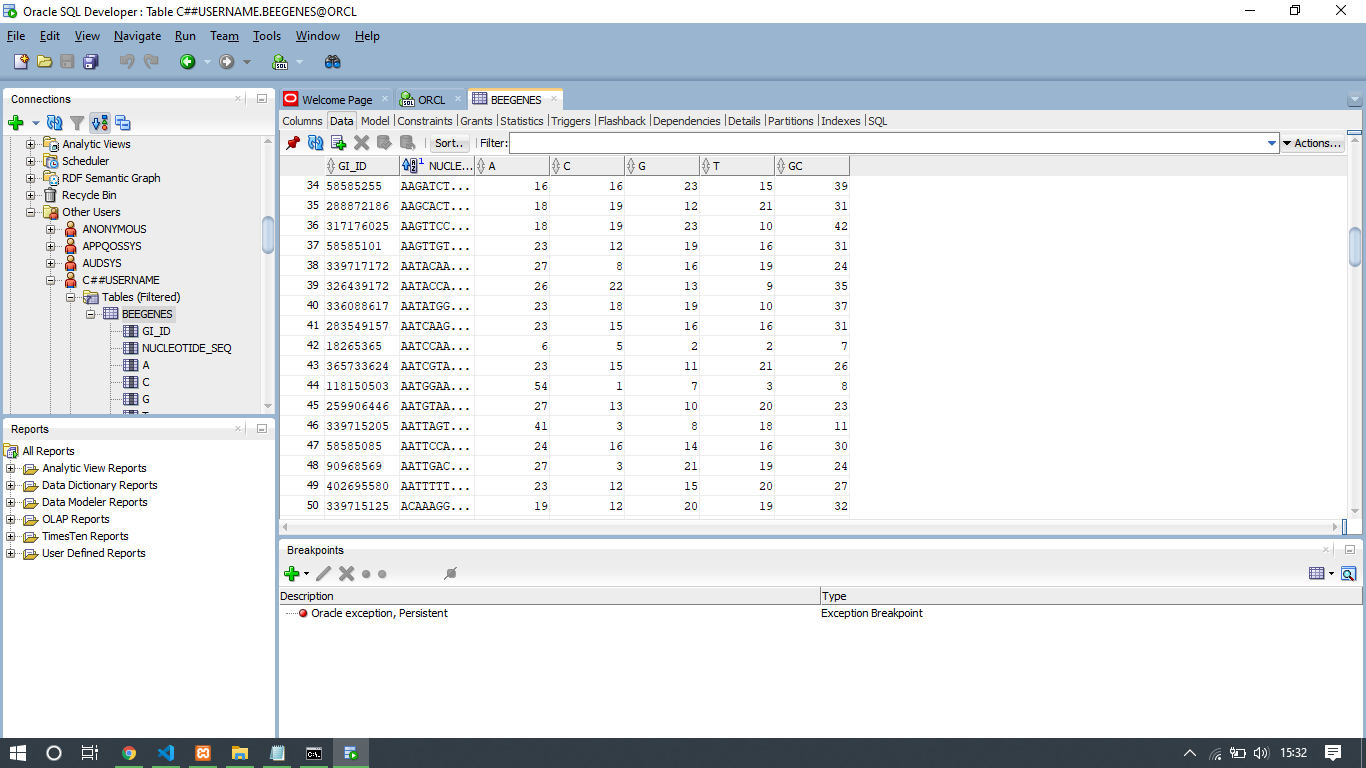
#create beegenes table

#please note credentials might be different for you and must be adjusted to run

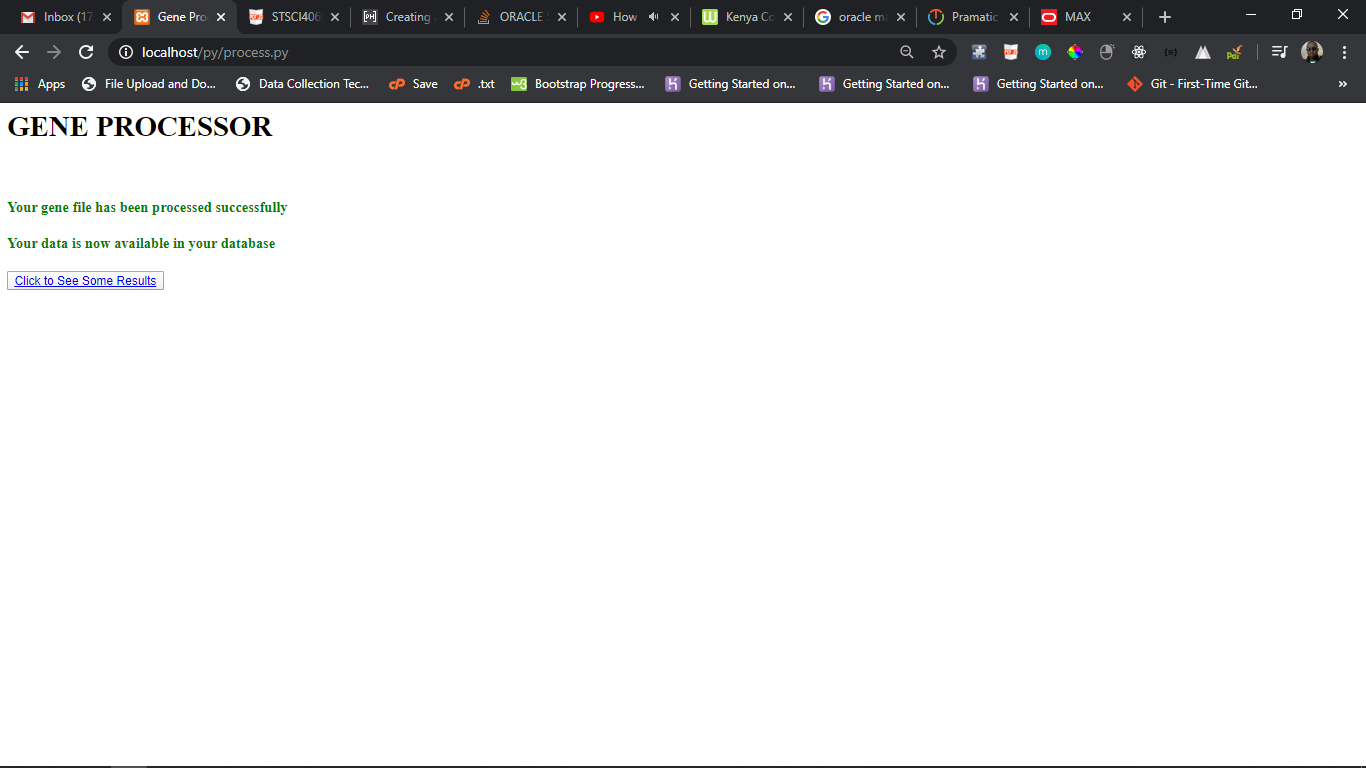
conn = cx\_Oracle.connect('c##username/password@localhost:1521/orcl')

cursor = conn.cursor()

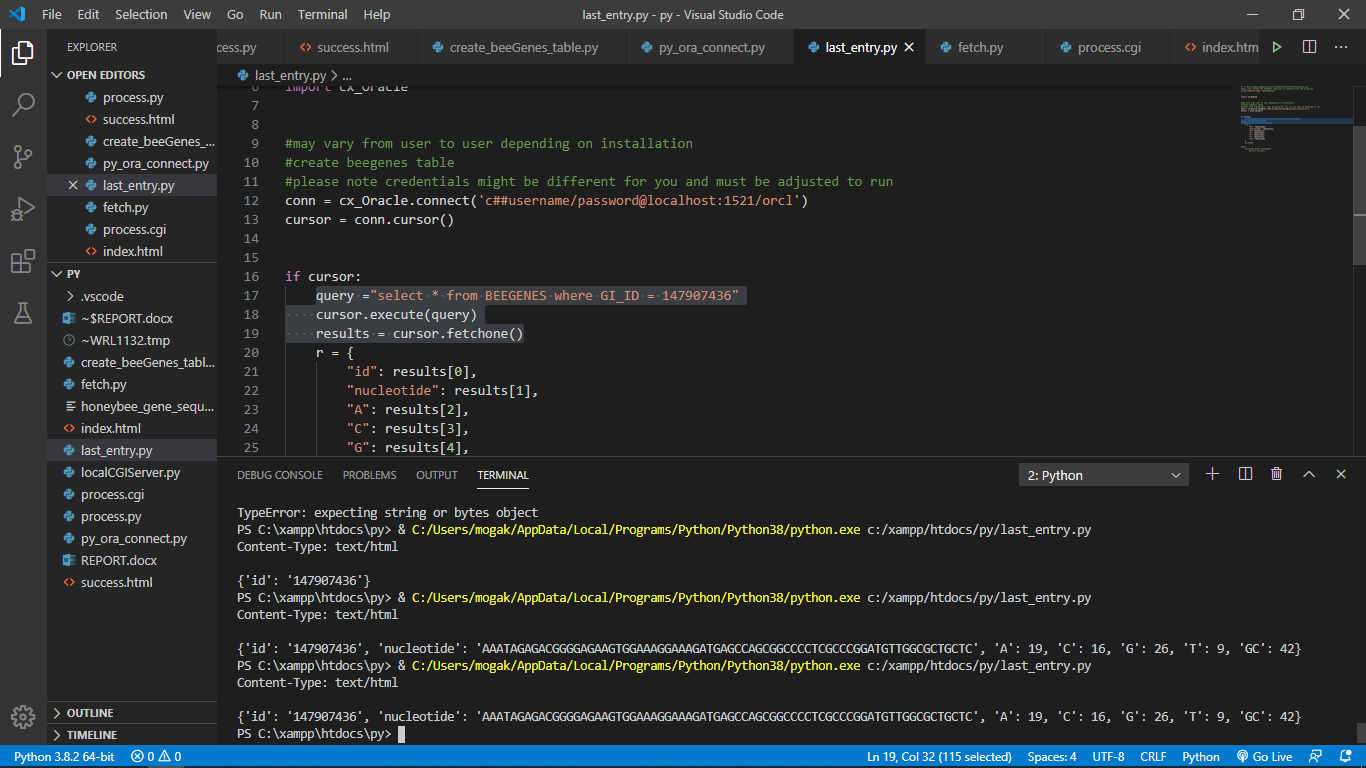
**6) PROCESSED DATA IN ORACLE DATABASE**



**7) SUCCESS DATA UPLOAD PAGE**



**8) QUERYING THE LAST ENTRY ON DATABASE SCRIPT:**



#! C:\Users\mogak\AppData\Local\Programs\Python\Python38\python.exe

#this code created the beeGenes table but is ommitted from the procee.py

print("Content-Type: text/html\n")

import cx\_Oracle

#may vary from user to user depending on installation

#create beegenes table

#please note credentials might be different for you and must be adjusted to run

conn = cx\_Oracle.connect('c##username/password@localhost:1521/orcl')

cursor = conn.cursor()

if cursor:

    query ="select \* from BEEGENES where GI\_ID = 147907436"

    cursor.execute(query)

    results = cursor.fetchone()

    r = {

        "id": results[0],

        "nucleotide": results[1],

        "A": results[2],

        "C": results[3],

        "G": results[4],

        "T": results[5],

        "GC": results[6]

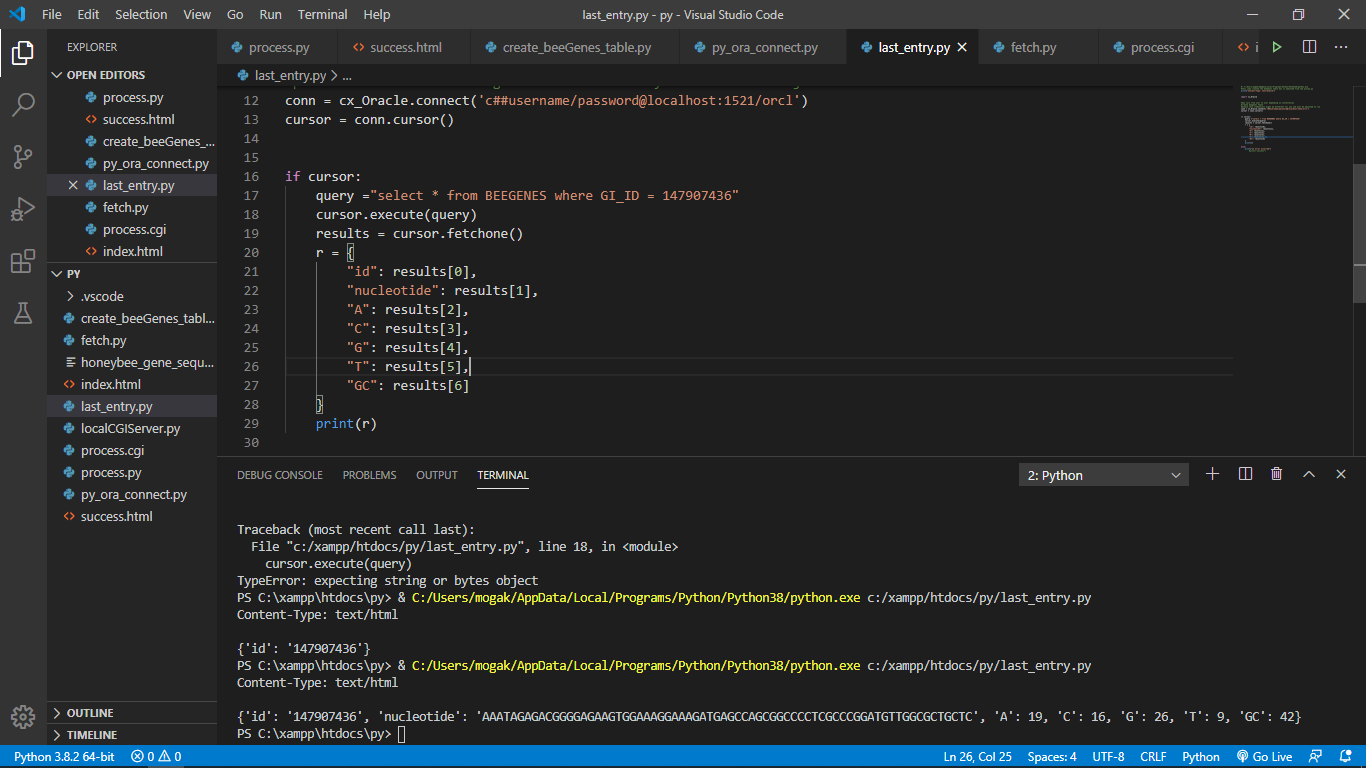
    }

    print(r)

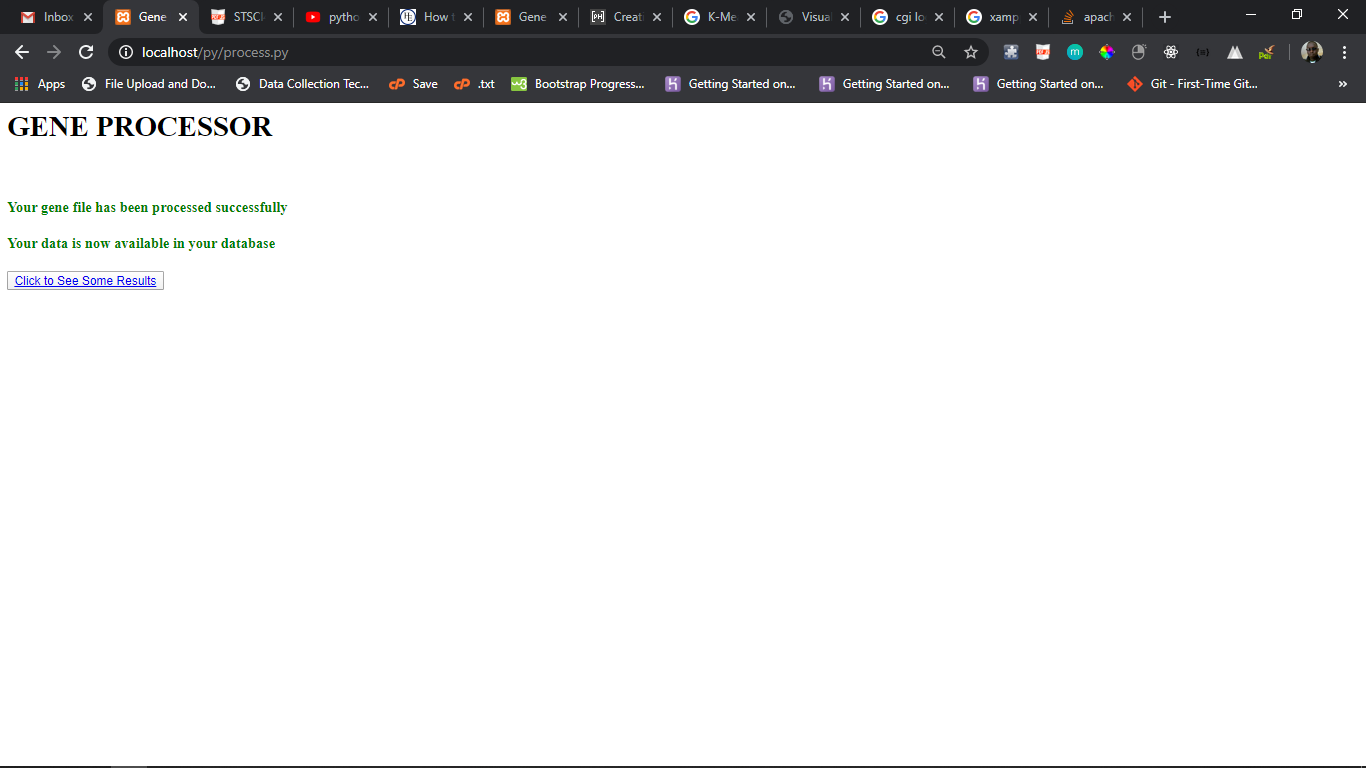
else:

    print("an error occurred")

        #print("success")



**9) VIEWING THE HIGHEST FREQUENCY DATA AS RETURNED AFTER CLICKING VIEW SOME DATA FROM THE SUCCESS PAGE**



**DATA RETURNED AFTER CLICKING SEE SOME DATA**

**CODE:**

#! C:\Users\mogak\AppData\Local\Programs\Python\Python38\python.exe

print("Content-Type: text/html\n")

#import Counter

import cx\_Oracle

#may vary from user to user depending on installation

#please note credentials might be different for you and must be adjusted to run

conn = cx\_Oracle.connect('c##username/password@localhost:1521/orcl')

cursor = conn.cursor()

#querying the db for gene data

#and returning sorted for user

def fetch\_data(connection):

    queries = ["select GI\_ID, A from BEEGENES where A = (select MAX(A) from BEEGENES)",

               "select GI\_ID, C from BEEGENES where C = (select MAX(C) from BEEGENES)",

               "select GI\_ID, G from BEEGENES where G = (select MAX(G) from BEEGENES)",

               "select GI\_ID, T from BEEGENES where T = (select MAX(T) from BEEGENES)"

               ]

    for query in queries:

        cursor.execute(query)

        results = cursor.fetchone()

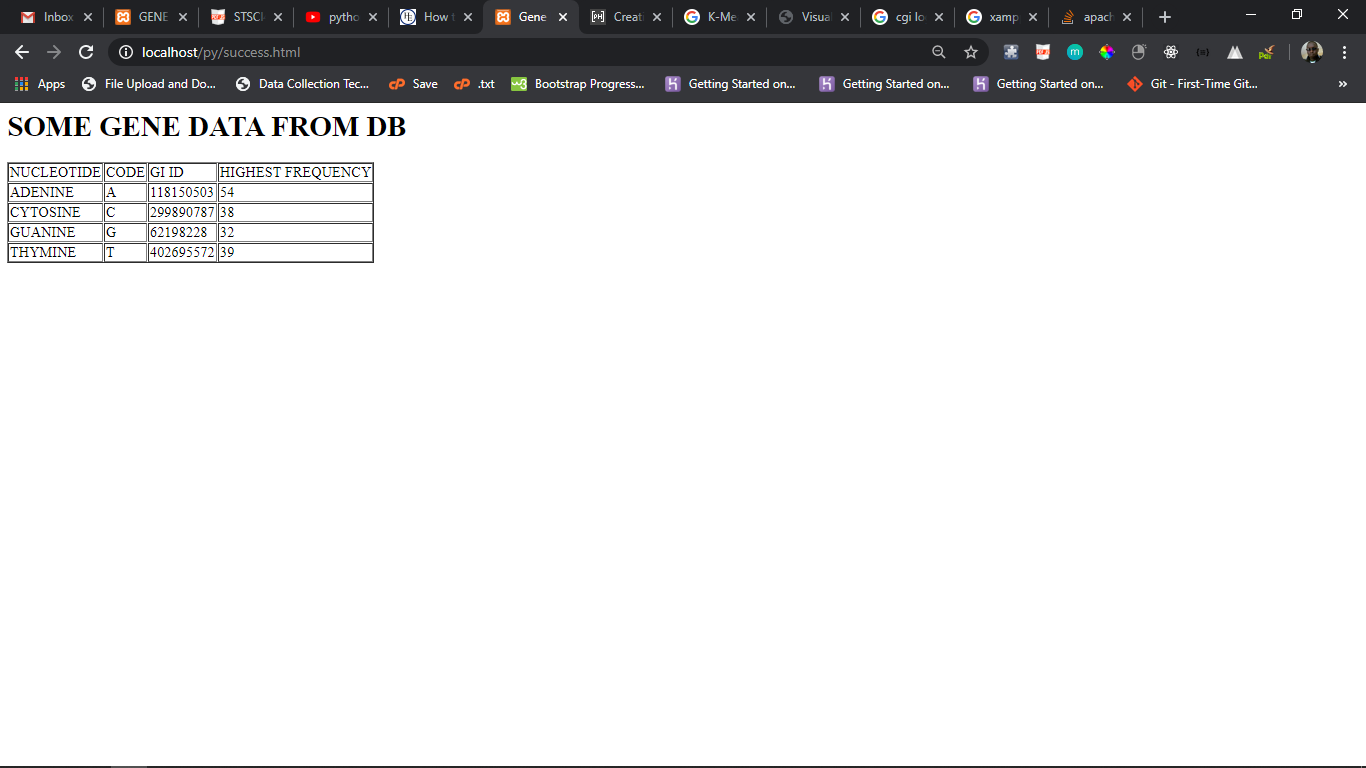
        r = {

            "id": results[0],

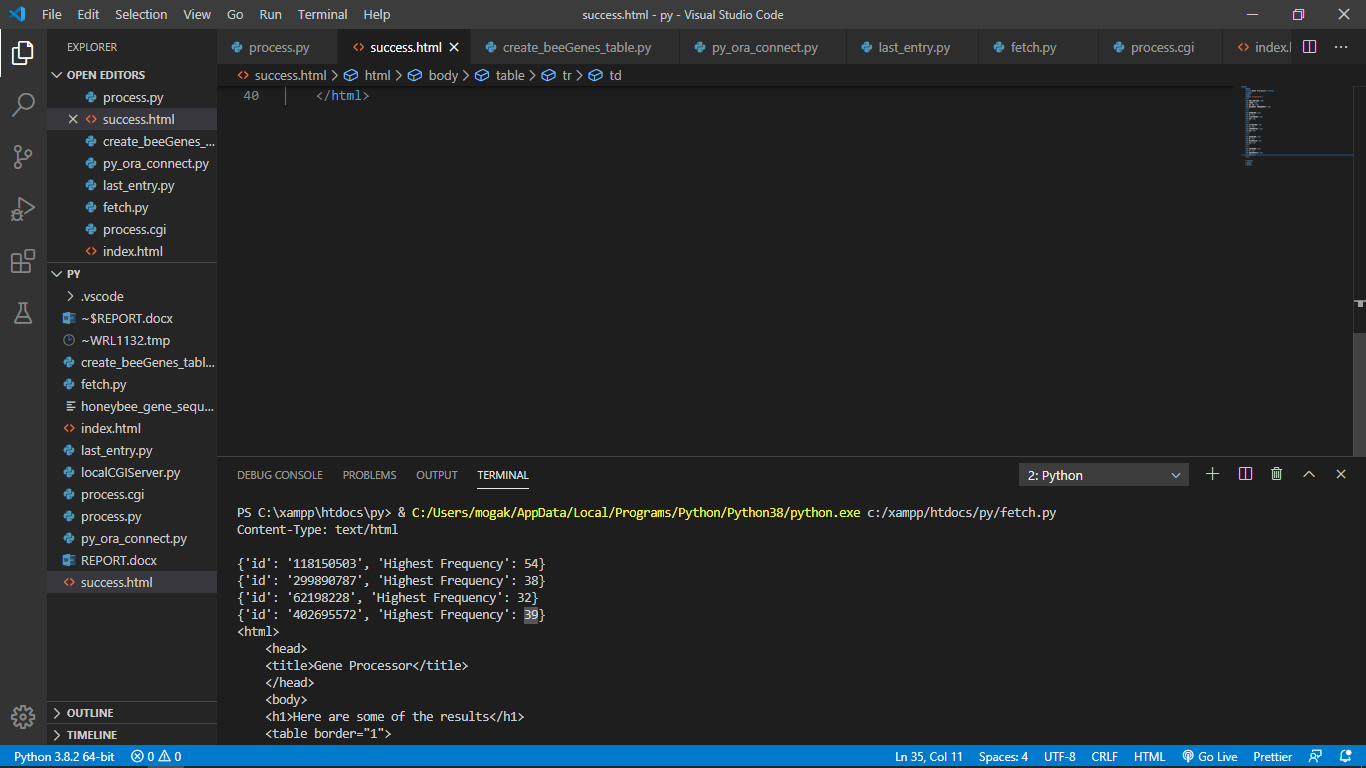
            "Highest Frequency": results[1]

        }

        print(r)



**RAW PYTHON DICTIONARY DATA FOR HIGHEST FREQUENCIES**



K-Means Analysis of frequencies A, T, and Relative frequency GC

Cell one importing of libraries

Cell two Jupiter Python and Oracle connection establishment

Cell three Data on frequencies

Script to be included in code folder

