DSCI 551 FINAL PROJECT REPORT

An insight into Climate Change using Emulated Distributed File System

---

Group Members and Background:

Shardul Nazirkar (USC ID - 6028439069) - B.Tech in Information Technology - Vishwakarma Institute of Information Technology, Pune, India

Niharika Abhange (USC ID - 1171034845) - B.Tech in Computer Science and Engineering - Maharashtra Institute of Technology, Pune, India

Nachiket Dunbray (USC ID - 7291924419) - B.Tech in Computer Science and Engineering - Maharashtra Institute of Technology

---

Project Description:

Climate Change is no longer a scientific theory. It is a real and devastating phenomenon that threatens life and our planet as we know it. The biggest reasons the circumstances keep deteriorating are rampant misinformation and lack of awareness and access to data. Some say it's a hoax but according to science and research, drastic changes in the environment have been reported in the last decade.

Every citizen of the world needs to comprehend the gravity of the situation before we reach an ultimatum. Statistics and Data are the most powerful tools to make the truth accessible to the public eye. This project will give in to the fact that climate change is real and has grave consequences, taking into consideration Datasets of the Global temperature, Sea water levels, and Ozone Layer.

Objectives:

1) To understand and design an EDFS, for different kinds of datasets based on Firebase and MySQL.

2) To implement partition based map and reduce as a part of the file handling system.
3) To correspond the file system to an application interface that is intuitive, well designed and allows the user to access the data seamlessly.
4) To optimize the system until it serves its purpose- creating awareness and providing accurate insights into the global climate scenario.

Workflow:

1) Construct Firebase-based emulation for a DFS.
2) Implementing partition based map and reduce (PMR) on data stored on EDFS, for the search or analytics functions in your application developed in the next task
3) Creating an app for searching the data stored in EDFS using the functions implemented using PMR.
4) Analytics operation to be performed on: Change in the different levels of the factors like ozone layer depletion, Temperature change, sea water level rise, etc.
5) Create an application interface that interactively exhibits different parameters that have been analyzed in step 4. Allow users to intuitively navigate through the dataset, and compare values based on locations and year, etc.

Datasets:

The following 3 datasets have been used in the course of this project:

1) Earth Surface Temperature: https://www.kaggle.com/datasets/berkeleyearth/climate-change-earth-surface-temperature-data

2) Sea Water Level https://www.star.nesdis.noaa.gov/socd/lsa/SeaLevelRise/LSA_SLR_timeseries.php

3) Ozone Layers - https://www.kaggle.com/datasets/jimschacko/ozone-layer

Note: These Datasets have been preprocessed before actual usage and hence cannot be used directly. Please refer to drive link to access the preprocessed datasets.

We aim to provide global insights into the effects of climate change. Hence datasets with ranges across the globe have been chosen. The availability of information for as long a time period as possible has been ensured.

Direct correlation to Climate change: The parameters of the datasets included- Earth Surface Temperature, Sea Water Levels and Ozone are directly correlated to climate change. These 3 parameters taken into consideration will provide a comprehensive insight into the quantitative effects of climate change.

IMPLEMENTATION OF TASK 1:

The main() function will be executed to create the initial structure of EDFS for every new instance. The namenode and datanode are created in this step. This command is implemented in Jupyter notebook.

CODE SNIPPETS FOR FIREBASE IMPLEMENTATION

MAIN

```
def main():
    global location
    r1 = requests.get('https://project-75c0c-default-rtdb.firebaseio.com/.json')
    dic = json.loads(r1.text)
    if ("datanode" not in dic.keys()):
      json_object1 = '{"datanode":""}'
      requests.patch('https://project-75c0c-default-rtdb.firebaseio.com/.json'.format(pwd),json_object1)
    if ("main" not in dic.keys()):
      json_object2 = '{"main":""}'
      requests.patch('https://project-75c0c-default-rtdb.firebaseio.com/.json'.format(pwd),json_object2)
    location = []
    location.append("main")
    while True:
        print("\n")
        for i in location:
          print(i,end="/")
        print("")
        command = input()
        command = command.split()
        main_command = command[0]
        if main_command == "mkdir":              #MKDIR
            try:
                filename = command[1]
                if filename.find("/")!=-1:
                  mkdir(str(filename))
                else:
                  mkdiring(location,str(filename))
        elif main_command == "ls":                #LS
            try:
              filename = command[1]
              ls(filename)
            except:
              lsing(location)
        elif main_command == "rm":                #RM
            try:
                filename = command[1]
                rm(str(filename),location)
            except:
                print("No file name")
```

```
        elif main_command == "cd":                  #CD
            try:
              directoryname = command[1]
              cd(location,str(directoryname))
            except:
              continue
        elif main_command=="cat":                   #CAT
          try:
              filename = command[1]
              cat(str(filename),location)
          except:
              continue
        elif main_command=="put":                   #PUT
          try:
              filename = command[1]
              partitions = command [2]
              put(str(filename),location,partitions)
          except:
              continue
        elif main_command=="getP":                   #getPartitions
          try:
              filename = command[1]
              df = getPartitions(str(filename),location)
              print(df)
          except:
              continue
        elif main_command=="readP":                   #readPartitions
          try:
              filename = command[1]
              partition = command [2]
              df = readPartitions(str(filename),location,partition)
              print(df)
        elif main_command =="exit":                 #EXIT
            break
        else:
            print("Command not found!!!!")
if __name__=="__main__":
    main()
```

Implementation of commands:

The mkdir command is executed to create a new directory, and it involves specifying the Block location. The mkdir function and its structure in the realtime database system are shown below.

A global variable named location also defined to store the location of the current directory that is being used.

MKDIR

```
pwd = ""
def mkdir(file_name):
  json_object = '{"'+file_name+'":""}'
  r1 = requests.get('https://project-75c0c-default-rtdb.firebaseio.com/.json')
  dic = json.loads(r1.text)
  if (file_name in dic.keys()):
    print("File already exists")
  else:
    r2 = requests.patch('https://project-75c0c-default-rtdb.firebaseio.com/.json'.format(pwd),json_object)

def mkdiring(location,file_name):
  prev_loc=""
  for i in location:
        prev_loc = prev_loc+"/"+i
  json_object = '{"'+prev_loc+'/'+file_name+'":""}'
  r1 = requests.get('https://project-75c0c-default-rtdb.firebaseio.com/.json')
  dic = json.loads(r1.text)
  if (file_name in dic.keys()):
    print("File already exists")
  else:
    r2 = requests.patch('https://project-75c0c-default-rtdb.firebaseio.com/.json'.format(pwd),json_object)
```

Following this are the screenshots and corresponding realtime database outputs of other EDFS commands like cd, ls, pwd, getchwd, put, and rm.

CD

```
def cd(location,file_name):
  global pwd

  if file_name == "..":
      location.pop()
      if len(location)==0:
        location.append("main")
        for i in location:
          print(i,end="/")
      else:
        for i in location:
          print(i,end="/")
  else:
      prev_loc = ""
      for i in location:
        prev_loc = prev_loc+"/"+i
      r1 = requests.get('https://project-75c0c-default-rtdb.firebaseio.com'+prev_loc+'.json')
      dic = json.loads(r1.text)
      if file_name in dic:
        location.append(file_name)
        for i in location:
          print(i,end="/")

      else:
        print("Location does not exist")
```

LS

```
def ls(file_name):
  r1 = requests.get('https://project-75c0c-default-rtdb.firebaseio.com/'+file_name+'.json')
  dic = json.loads(r1.text)
  for i in dic.keys():
        print(i,end=" ")
def lsing(location):
  prev_loc=""
  for i in location:
        prev_loc = prev_loc+"/"+i
  r1 = requests.get('https://project-75c0c-default-rtdb.firebaseio.com'+prev_loc+'.json')
  dic = json.loads(r1.text)
  for i in dic.keys():
        print(i,end =" ")
```

+ Code + Text

RM

```
def rm(file_name,location):
  json_object = '{"'+file_name+'":""}'
  prev_loc = ""
  for i in location:
        prev_loc = prev_loc+"/"+i

  r1 = requests.get('https://project-75c0c-default-rtdb.firebaseio.com'+prev_loc+'.json')
  dic = json.loads(r1.text)
  partition_locs = getPartitions(file_name,location)
  for i in partition_locs:
    r2 = requests.delete(i)
  r3 = requests.delete('https://project-75c0c-default-rtdb.firebaseio.com'+prev_loc+"/"+file_name+'.json')
  if(r2.status_code!=200):
    print("File doesn't exists")
```

CAT

```
[6] def cat(file_name,location):
      prev_loc = ""
      df = []
      for i in location:

            prev_loc = prev_loc+"/"+i
      location_list=getPartitions(file_name,location)
      for i in range(1,len(location_list)+1):
        df.append(readPartitions(file_name,location,i))

      maindf = pd.concat(df)
      print(maindf)
```

PUT

```
def put(file,location,k):
  df = pd.read_csv(file)
  files = file.split(".")
  file = files[0]
  count_row = df.shape[0]
  k = int(k)
  sliceval = round(count_row/k)
  partitions_df=[]
  z=0
  for i in range (0,k):
    partitions_df.append(df[z:z+sliceval])
    z=z+sliceval
  file_location = []
  for i in range (0,len(partitions_df)):
    js = partitions_df[i].to_json()
    diction = json.loads(js)
    jstring = json.dumps(diction)
    r1 = requests.patch('https://project-75c0c-default-rtdb.firebaseio.com/datanode/'+str(i+1)+'/'+file+str(i+1)+'.json',jstring)
    file_location.append('https://project-75c0c-default-rtdb.firebaseio.com/datanode/'+str(i+1)+'/'+file+str(i+1)+'.json')


  prev_loc = ""
  metadiction = {"File Name":file,"Number of Partitions":k,"Locations":file_location}
  metajstring = json.dumps(metadiction)
  for i in location:
    prev_loc = prev_loc+"/"+i
  r2 = requests.patch('https://project-75c0c-default-rtdb.firebaseio.com'+prev_loc+'/'+file+'.json',metajstring)
```

The next function getPartitions() function gives us the locations od the file partitions.

getPartitions

```
def getPartitions(file_name,location):
  json_object = '{"'+file_name+'":""}'
  prev_loc = ""
  for i in location:
    prev_loc = prev_loc+"/"+i
  r1 = requests.get('https://project-75c0c-default-rtdb.firebaseio.com'+prev_loc+"/"+file_name+'.json')
  dic = json.loads(r1.text)
  location_list=[]
  location_list=dic["Locations"]
  return location_list
```

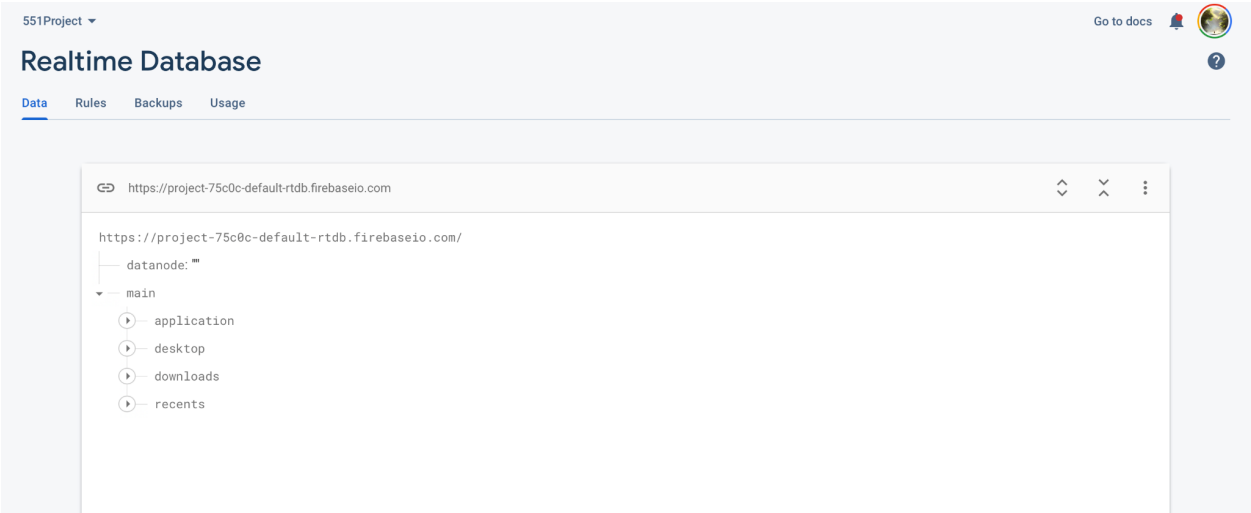
After getPartitions() the actual contents of the file are read by the readPartitions() function. The implementation of readPartition(file, partition#) which returns the content of partition of the specified file is given below.

readPartitions

```
def readPartitions(file_name,location,kth):
  json_object = '{"'+file_name+'":""}'
  prev_loc = ""
  kth = int(kth)
  for i in location:
    prev_loc = prev_loc+"/"+i
  location_list = getPartitions(file_name,location)
  if(kth <= len(location_list)):
    r1 = requests.get(location_list[kth-1])
    dic = json.loads(r1.text)
    df=pd.DataFrame(dic)
    return df
  else:
    print("Required partition does not exist!!!")
```

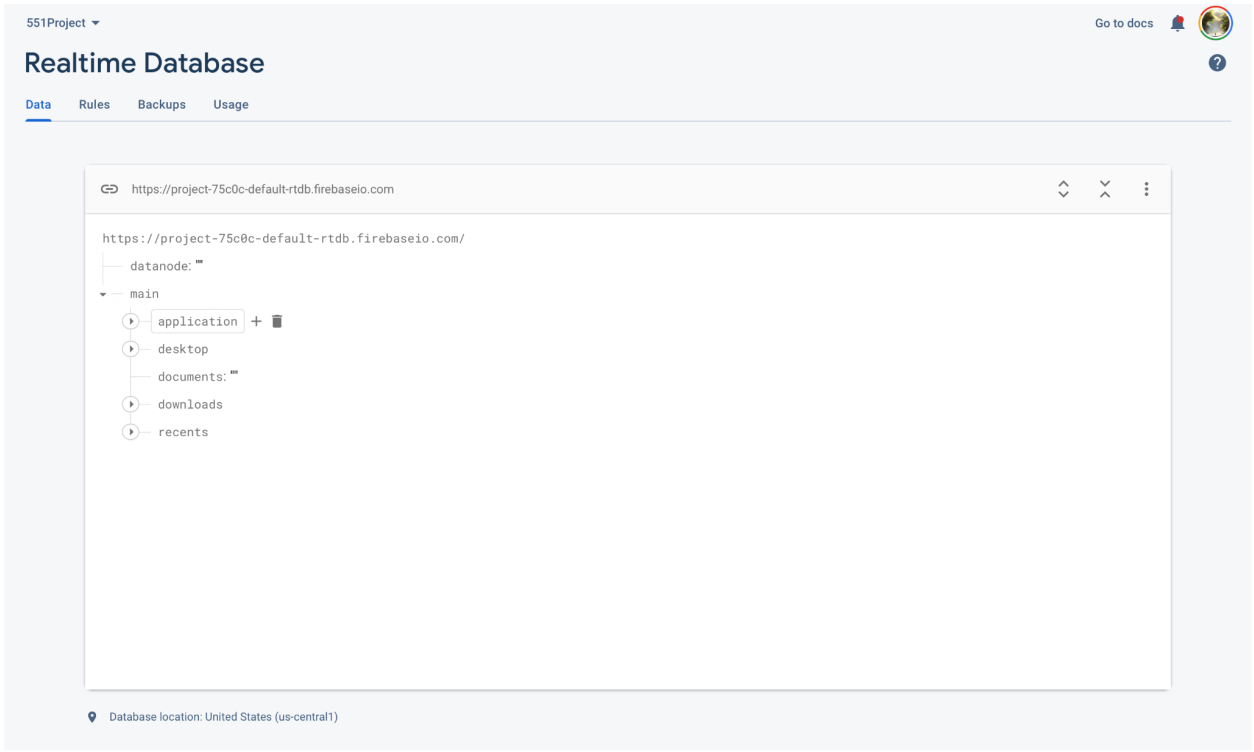
OUTPUTS:

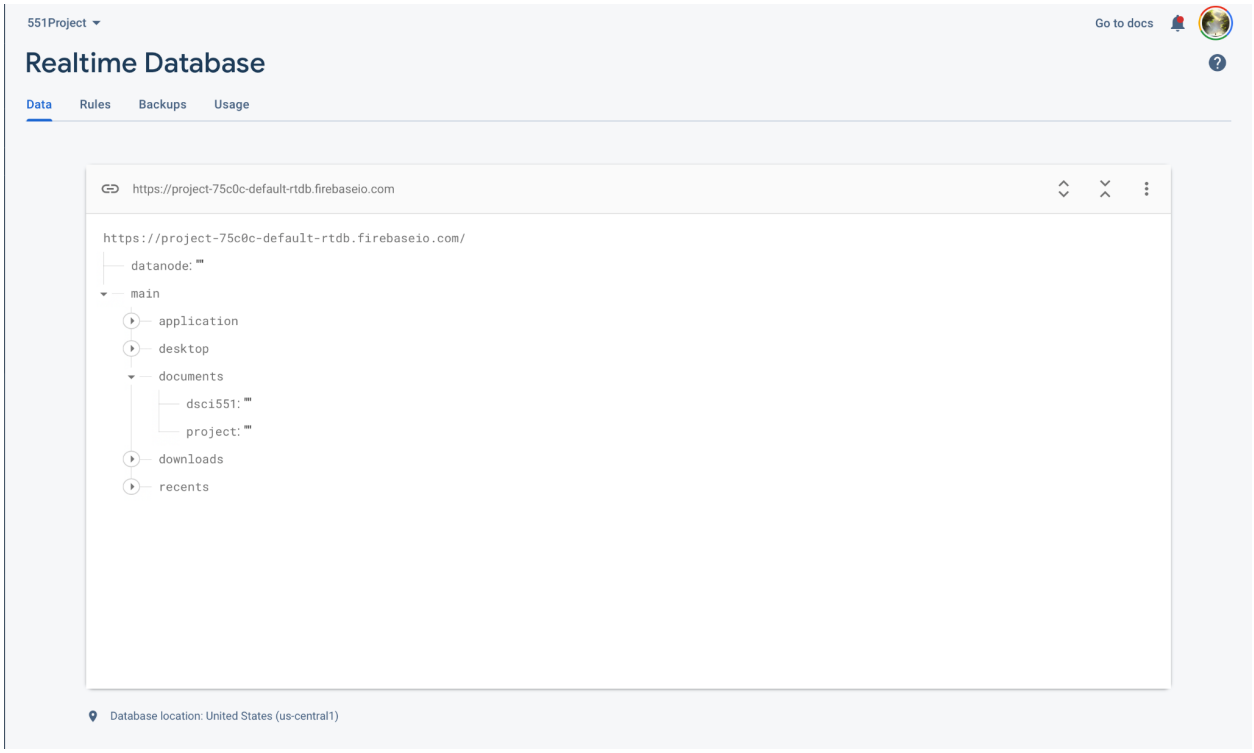
Main Structure of System



Implementing the CLI commands

```
...

    main/
    ls
    application desktop documents downloads recents

    main/
    mkdir documents


    main/
    cd documents
    main/documents/

    main/documents/
    mkdir dsci551


    main/documents/
    mkdir project


    main/documents/
    cd project
    main/documents/project/

    main/documents/project/
    put Ozone.csv 4
```


551Project
Go to docs
Realtime Database
Data
Rules
Backups
Usage
https://project-75c0c-default-rtdb.firebaseio.com
https://project-75c0c-default-rtdb.firebaseio.com/
datanode: ""
main
application
desktop
documents: ""
downloads
recents
Database location: United States (us-central1)


551Project
Go to docs
Realtime Database
Data
Rules
Backups
Usage
https://project-75c0c-default-rtdb.firebaseio.com
https://project-75c0c-default-rtdb.firebaseio.com/
datanode: ""
main
application
desktop
documents
dsci551: ""
project: ""
downloads
recents
Database location: United States (us-central1)

https://project-75c0c-default-rtdb.firebaseio.com/

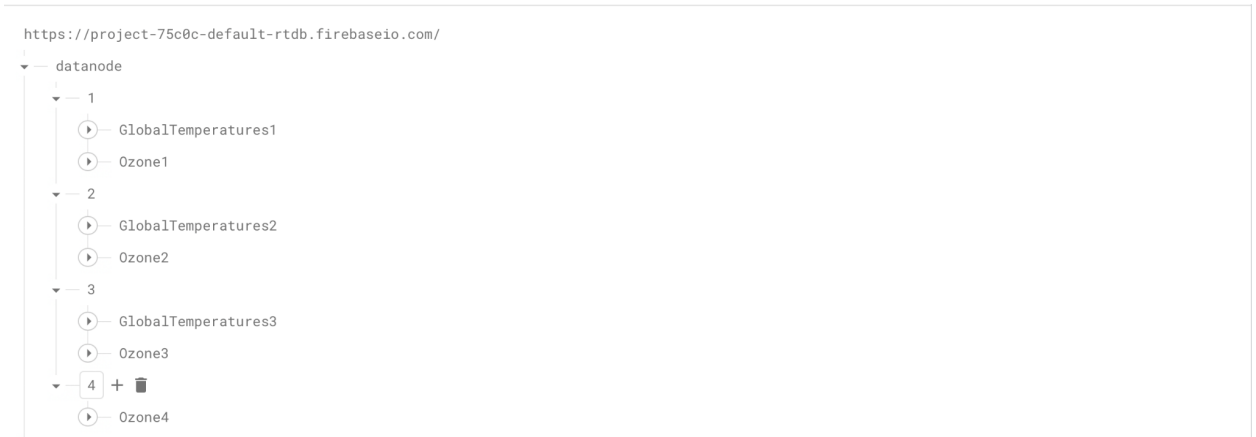
- datanode
  - 1
    - Ozone1
  - 2
    - Ozone2
  - 3
    - Ozone3
  - 4
    - Ozone4
- main
  - application
  - desktop
  - documents
    - dsci551: ""
    - project
      - Ozone
        - File Name: "Ozone"
        - Locations
          - 0: "https://project-75c0c-default-rtdb.firebaseio.com/datanode/1/Ozone1.json"
          - 1: "https://project-75c0c-default-rtdb.firebaseio.com/datanode/2/Ozone2.json"
          - 2: "https://project-75c0c-default-rtdb.firebaseio.com/datanode/3/Ozone3.json"
          - 3: "https://project-75c0c-default-rtdb.firebaseio.com/datanode/4/Ozone4.json"
        - Number of Partitions: 4
  - downloads
  - recents

```
...
    main/documents/project/
    put GlobalTemperatures.csv 3


    main/documents/project/
    cat Ozone
          Class   V1  V10  V11  V12  V13  V14  V15  V16  V17  ...          V67  \
    0       1.0  0.8  2.3  3.7  5.5  5.1  5.4  5.4  4.7  4.3  ...  5795.000000
    1       1.0  2.8  3.4  4.2  4.5  4.5  4.3  5.5  5.1  3.8  ...  5805.000000
    2       1.0  2.9  2.5  3.1  4.0  4.4  4.6  5.6  5.4  5.2  ...  5790.000000
    3       1.0  4.7  3.1  3.3  3.1  2.3  2.1  2.2  3.8  2.8  ...  5775.000000
    4       1.0  2.6  1.4  2.2  2.0  3.0  3.0  3.1  3.1  2.7  ...  5818.821222
    ...     ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...          ...
    2529    1.0  0.3  2.2  2.8  2.8  2.8  3.4  3.9  3.8  3.5  ...  5800.000000
    2530    1.0  1.0  2.4  4.2  4.6  4.2  4.8  5.0  4.7  3.7  ...  5845.000000
    2531    1.0  0.8  1.5  2.9  3.3  3.7  3.9  3.6  3.7  3.3  ...  5845.000000
    2532    1.0  1.3  2.0  2.5  2.1  2.5  3.6  3.9  4.0  3.6  ...  5845.000000
    2533    1.0  1.5  3.0  3.6  3.2  3.3  4.0  3.1  3.4  3.8  ...  5820.000000

                V68        V69   V7           V70        V71   V72   V8   V9  \
    0    -12.100000  17.900000  1.5  10330.000000 -55.000000  0.00  1.7  1.9
    1     14.050000  29.000000  2.9  10275.000000 -55.000000  0.00  2.8  3.1
    2     17.900000  41.300000  2.5  10235.000000 -40.000000  0.00  2.7  2.2
    3     31.150000  51.700000  2.8  10195.000000 -40.000000  2.08  2.5  2.4
    4     10.511051  37.388335  1.2  10164.198442  -0.119949  0.58  1.4  1.3
    ...         ...        ...  ...           ...        ...   ...  ...  ...
    2529 -25.600000  21.800000  0.4  10295.000000  65.000000  0.00  0.4  1.3
    2530 -19.400000  19.100000  1.8  10310.000000  15.000000  0.00  1.5  2.1
    2531  -9.600000  35.200000  0.8  10275.000000 -35.000000  0.00  1.1  1.5
    2532 -19.600000  34.200000  1.1  10245.000000 -30.000000  0.05  1.0  1.9
    2533   1.950000  39.350000  1.6  10220.000000 -25.000000  0.00  1.4  1.6

              id
```

```
...          ...         ...  ...  ...           ...          ...   ...  ...  ...
2529 -25.600000  21.800000  0.4  10295.000000  65.000000  0.00  0.4  1.3
2530 -19.400000  19.100000  1.8  10310.000000  15.000000  0.00  1.5  2.1
2531  -9.600000  35.200000  0.8  10275.000000 -35.000000  0.00  1.1  1.5
2532 -19.600000  34.200000  1.1  10245.000000 -30.000000  0.05  1.0  1.9
2533   1.950000  39.350000  1.6  10220.000000 -25.000000  0.00  1.4  1.6

          id
0        1.0
1        2.0
2        3.0
3        4.0
4        5.0
...      ...
2529  2530.0
2530  2531.0
2531  2532.0
2532  2533.0
2533  2534.0

[3168 rows x 74 columns]
```

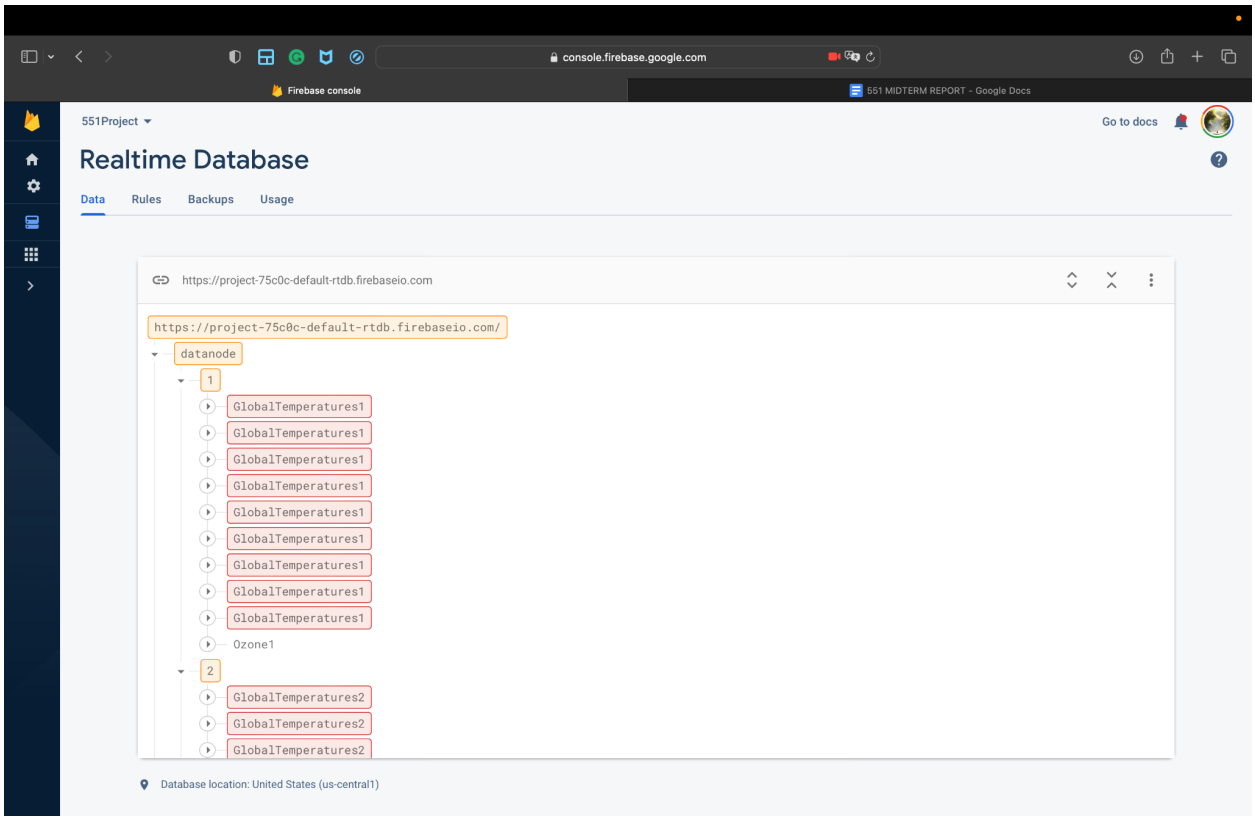
```
...
main/documents/project/
getP Ozone
['https://project-75c0c-default-rtdb.firebaseio.com/datanode/1/Ozone1.json', 'https://project-75c0c-default-rtdb.firebaseio.com/datanode/2/Ozor
```

```
...  main/documents/project/
readP Ozone 3
      Class   V1  V10  V11  V12  V13  V14  V15  V16  V17  ...     V67    V68  \
1268      1  0.5  3.4  2.5  2.5  1.7  1.8  1.7  1.3  0.7  ...  5915.0  33.70
1269      1  1.2  2.9  2.1  1.1  1.5  1.1  0.8  0.7  1.7  ...  5910.0  36.45
1270      1  1.2  1.1  0.8  0.8  1.3  1.8  2.6  4.3  2.6  ...  5915.0  32.55
1271      1  0.9  1.8  1.9  1.1  1.4  1.6  2.0  2.1  3.5  ...  5935.0  28.20
1272      1  1.4  2.8  2.5  1.9  2.0  2.1  2.1  2.2  1.7  ...  5920.0  29.00
...     ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...     ...    ...
1897      1  2.8  3.7  4.1  4.2  3.0  1.1  0.7  2.4  2.9  ...  5785.0  13.90
1898      1  0.9  1.8  2.3  2.4  2.2  2.8  3.2  4.1  3.8  ...  5765.0  32.75
1899      1  3.0  4.0  3.8  3.7  3.4  3.2  2.7  2.5  2.6  ...  5790.0  22.05
1900      1  0.4  2.2  2.2  2.6  2.3  1.9  2.4  3.0  2.4  ...  5795.0  26.45
1901      1  1.9  5.7  5.7  5.5  4.9  5.5  5.9  5.6  6.0  ...  5790.0  27.05

        V69   V7      V70   V71   V72   V8        V9    id
1268  48.20  0.8  10180.0 -20.0  0.00  2.4  3.400000  1269
1269  49.25  0.4  10165.0 -15.0  0.00  1.6  2.800000  1270
1270  44.50  0.5  10175.0  10.0  0.00  0.7  0.900000  1271
1271  41.10  0.0  10180.0   5.0  0.00  1.3  1.400000  1272
1272  41.90  0.2  10160.0 -20.0  0.00  1.3  3.000000  1273
...     ...  ...      ...   ...   ...  ...       ...   ...
1897  45.50  2.2  10130.0 -60.0  0.00  2.8  2.539037  1898
1898  50.25  1.4  10115.0 -15.0  0.00  1.1  1.900000  1899
1899  52.25  3.8  10110.0  -5.0  0.08  3.6  4.300000  1900
1900  52.60  1.9  10080.0 -30.0  0.03  2.1  2.100000  1901
1901  47.05  5.2  10135.0  55.0  0.00  5.1  5.500000  1902

[634 rows x 74 columns]
```



- main
  - application
  - desktop
  - documents
    - dsci551: ""
    - project
      - GlobalTemperatures
        - File Name: "GlobalTemperatures"
        - Locations
        - Number of Partitions: 3
      - Ozone
        - File Name: "Ozone"
        - Locations
        - Number of Partitions: 4
  - downloads
  - recents

```
main/documents/project/
rm GlobalTemperatures
```



CODE SNIPPETS FOR MySQL IMPLEMENTATION:

mkdir

In [10]:
```
def mkdir(name):
    mycursor.execute("insert into main (filename, parent, type) values (\""+ name +"\", \""+location[-1]+"\",'dir')")
    conn.commit()
```

LS

In [11]:
```
def ls(location):
    mycursor.execute("select filename from main where \""+ location[-1] +"\" = parent order by filename asc")
    str=""
    for x in mycursor:
        str=str + x[0] + " "
    print(str)
    return str
```

PUT

In [15]:
```
def put(file,location,k):
    partnames = []
    df = pd.read_csv(file)
    files = file.split(".")
    file = files[0]
    count_row = df.shape[0]
    k = int(k)
    sliceval = round(count_row/k)
    partitions_df=[]
    z=0
    for i in range (0,k):
        partitions_df.append(df[z:z+sliceval])
        z=z+sliceval

    for i in range(0,len(partitions_df)):
        data=partitions_df[i]
        tabname = str(file+str(i+1))
        data.to_sql(name=tabname, con=engine, if_exists = 'replace', index=False)
        partnames.append(tabname)
    partnames = ','.join(partnames)
    k = str(k)
    mycursor.execute("replace into metadata values (\""+ file +"\",\""+ partnames +"\",\""+ k +"\")")
    mycursor.execute("replace into main (filename, parent, type) values (\""+ file +"\",\""+ location[-1] +"\",'file')")
    conn.commit()
```

RM

In [13]:
```
def rm(file_name,location):
        x = mycursor.execute("select type from main where filename = \""+ file_name +"\"")
        for x in mycursor:
            if (x[0]) == "file":
                tabnames = getPartitions(file_name)
                mycursor.execute("delete from metadata where filename =\""+ file_name +"\"")

                locats = list(tabnames.split(","))
                for i in locats:
                    mycursor.execute("drop table "+i)

        mycursor.execute("delete from main where filename =\""+ file_name +"\" and parent = \""+ location[-1] +"\"")
        mycursor.execute("delete from main where parent = \""+ file_name +"\"")
        mycursor.execute("select type from main where filename = \""+ file_name +"\"")
        conn.commit()
        return("File deleted!")
```

getPartitions

In [16]:
```
def getPartitions(filename):
    df = pd.read_sql("select locations from metadata where filename =\""+ filename +"\"",conn)
    parts = df["locations"][0]
    print(type(parts))
    return(parts)
```

readPartitions

In [18]:
```
def readPartitions(filename,kth):
    kth = int(kth)
    locats = getPartitions(filename)
    locats = list(locats.split(","))
    if len(locats)>=kth and kth !=0:
        stringer = locats[kth-1]
        df = pd.read_sql("select * from "+ stringer,conn)
        return df
    else:
        print("No such partition")
```

CAT

In [14]:
```
def cat(filename,location):
    newdf = []
    locats = getPartitions(filename)
    locats = list(locats.split(","))
    #print(locats)
    for i in range (1,len(locats)+1):
        df = readPartitions(filename,i)
        newdf.append(df)
    maindf = pd.concat(newdf, ignore_index=True)
    return maindf
```

MySQL DATABASE STRUCTURE

fileid	filename	parent	type
1	desktop	root	dir
2	documents	root	dir
3	applications	root	dir
4	files	root	dir
5	weather	files	dir
6	Ozone	weather	file
8	CityTemp	weather	file
NULL	NULL	NULL	NULL

Tables_in_project
citytemp1
citytemp2
citytemp3
citytemp4
main
metadata
ozone1
ozone2
ozone3

filename	locations	noparts
Ozone	Ozone1,Ozone2,Ozone3	3
CityTemp	CityTemp1,CityTemp2,CityTemp3,CityTemp4	4

Analytics and Search operations using Partition Map Reduce

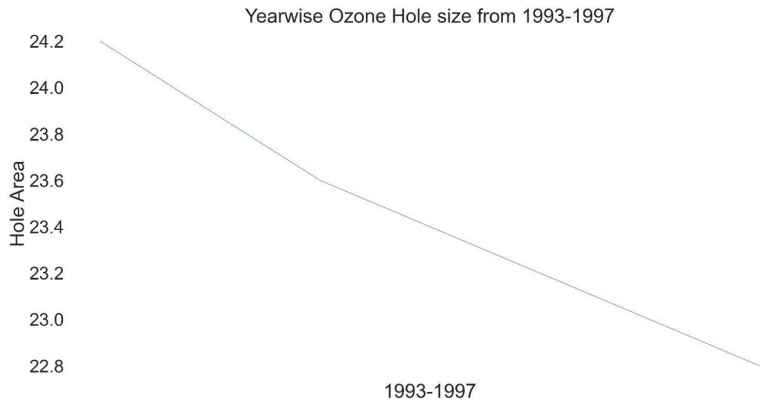
The task 2 is the final implementation of our project.
Implementation of Partition based map and reduce on data stored on EDFS, for search and analytics

- The map function takes input as the number of partitions and gives all the partitions as output.
- The reduce function takes each partition and gives a single output after combining them.

CODE SNIPPETS FOR SEARCH AND ANALYTICS USING PMR:

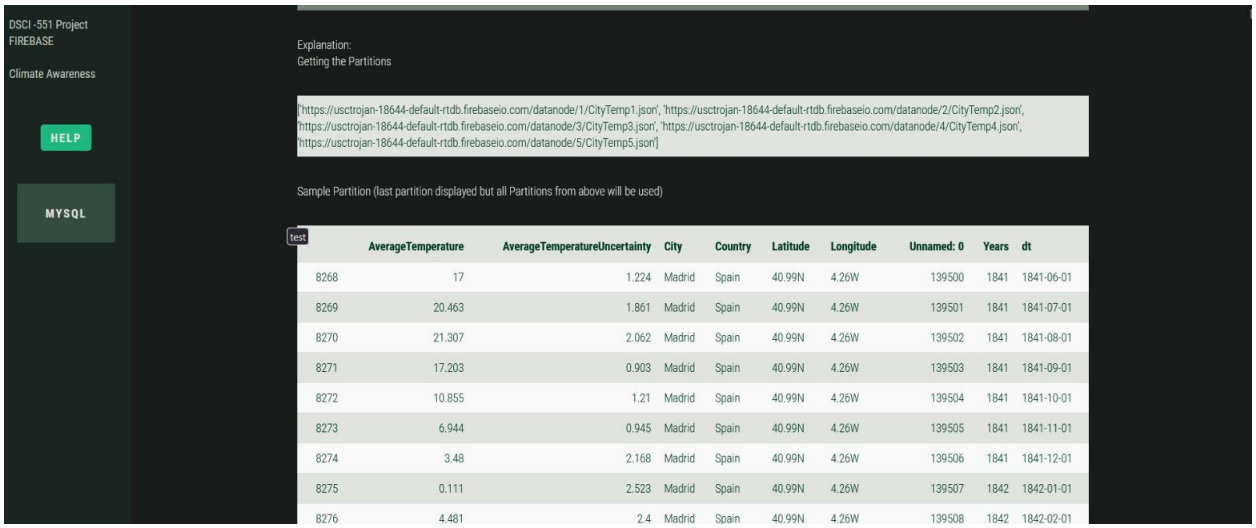
Analytics - Ozone

In [21]:
```
def ozograph(location, start_year, end_year):
    parts = getPartitions("Ozone")
    ran = len(parts)
    start_year = int(start_year)
    end_year = int(end_year)
    index = 0
    for i in range(1, len(parts.split(','))+1):
        #
        ozo = readPartitions("Ozone" , i)
        for i in range (0, len(ozo)):
            if start_year == ozo['Year'][i]:
                si = index
            if end_year == ozo['Year'][i]:
                ei = index
            index = index + 1
    temu = cat("Ozone",location).dropna()
    sns.set(rc={'figure.figsize':(40,20)})
    sns.set(font_scale=5)
    ax = sns.lineplot(data= temu.iloc[si:ei],x="Year", y="Hole Area")
    ax.set_xticks([])
    ax.set(xlabel=str(start_year) + '-' + str(end_year),
           ylabel='Hole Area',
           title='Yearwise Ozone Hole size from ' + str(start_year) + '-' + str(end_year))
    return anvil.mpl_util.plot_image()
```



Search - Country Temp

In [23]:
```
def mapsearchTemp(country, temp):
    return(temp[temp['Country'] == country])

def reduceTemp(country):
    parts = getPartitions("CityTemp")
    outputs = []
    for i in range (1, len(parts.split(','))+1):
        temp = readPartitions("CityTemp" , i).drop('Unnamed: 0', axis = 1)
        outputs.append(mapsearchTemp(country, temp))
    return parts,temp,outputs[-1],pd.concat(outputs).dropna()
```



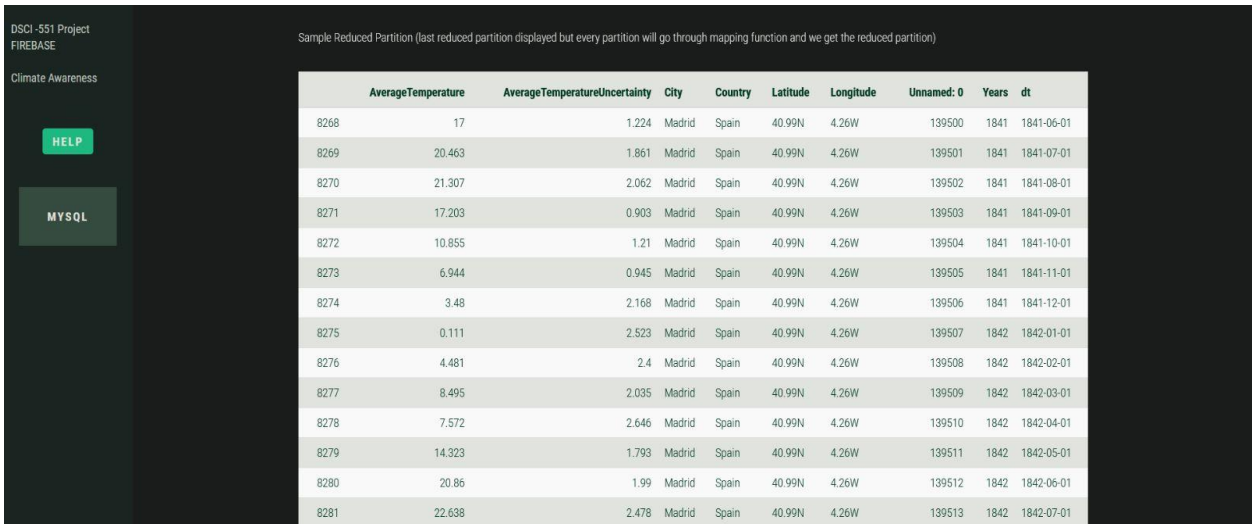
Explanation:
Getting the Partitions

['https://usctrojan-18644-default-rtdb.firebaseio.com/datanode/1/CityTemp1.json', 'https://usctrojan-18644-default-rtdb.firebaseio.com/datanode/2/CityTemp2.json', 'https://usctrojan-18644-default-rtdb.firebaseio.com/datanode/3/CityTemp3.json', 'https://usctrojan-18644-default-rtdb.firebaseio.com/datanode/4/CityTemp4.json', 'https://usctrojan-18644-default-rtdb.firebaseio.com/datanode/5/CityTemp5.json']

Sample Partition (last partition displayed but all Partitions from above will be used)

test

	AverageTemperature	AverageTemperatureUncertainty	City	Country	Latitude	Longitude	Unnamed: 0	Years	dt
8268	17	1.224	Madrid	Spain	40.99N	4.26W	139500	1841	1841-06-01
8269	20.463	1.861	Madrid	Spain	40.99N	4.26W	139501	1841	1841-07-01
8270	21.307	2.062	Madrid	Spain	40.99N	4.26W	139502	1841	1841-08-01
8271	17.203	0.903	Madrid	Spain	40.99N	4.26W	139503	1841	1841-09-01
8272	10.855	1.21	Madrid	Spain	40.99N	4.26W	139504	1841	1841-10-01
8273	6.944	0.945	Madrid	Spain	40.99N	4.26W	139505	1841	1841-11-01
8274	3.48	2.168	Madrid	Spain	40.99N	4.26W	139506	1841	1841-12-01
8275	0.111	2.523	Madrid	Spain	40.99N	4.26W	139507	1842	1842-01-01
8276	4.481	2.4	Madrid	Spain	40.99N	4.26W	139508	1842	1842-02-01

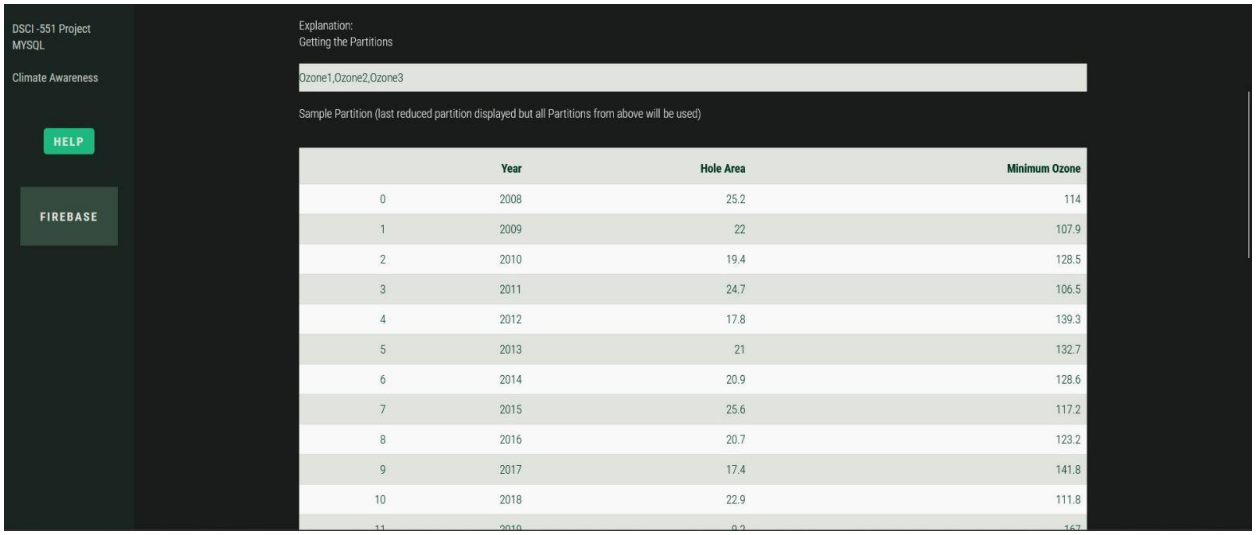


Sample Reduced Partition (last reduced partition displayed but every partition will go through mapping function and we get the reduced partition)

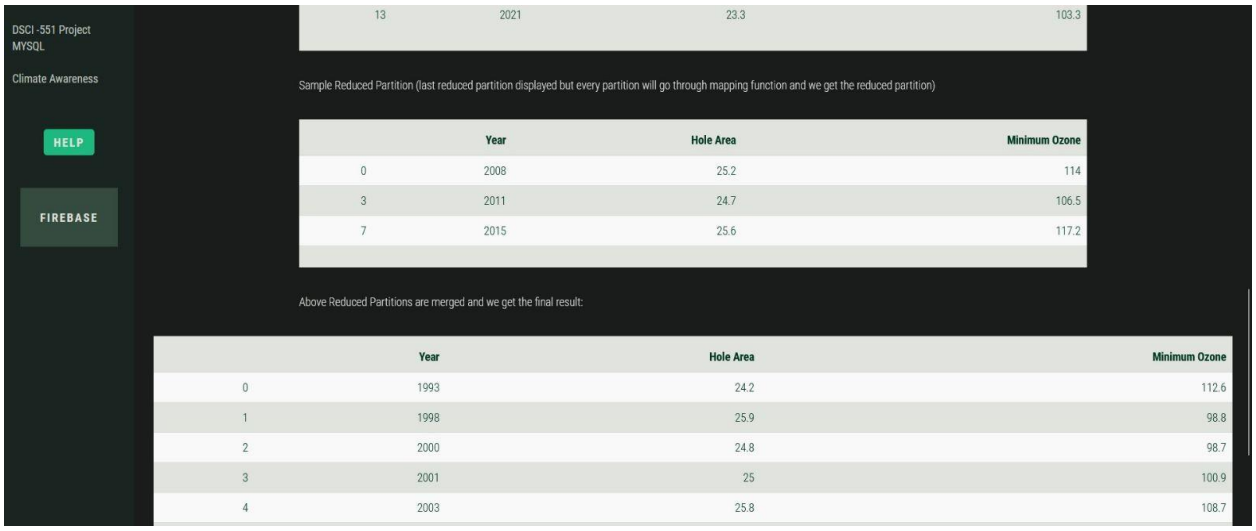
	AverageTemperature	AverageTemperatureUncertainty	City	Country	Latitude	Longitude	Unnamed: 0	Years	dt
8268	17	1.224	Madrid	Spain	40.99N	4.26W	139500	1841	1841-06-01
8269	20.463	1.861	Madrid	Spain	40.99N	4.26W	139501	1841	1841-07-01
8270	21.307	2.062	Madrid	Spain	40.99N	4.26W	139502	1841	1841-08-01
8271	17.203	0.903	Madrid	Spain	40.99N	4.26W	139503	1841	1841-09-01
8272	10.855	1.21	Madrid	Spain	40.99N	4.26W	139504	1841	1841-10-01
8273	6.944	0.945	Madrid	Spain	40.99N	4.26W	139505	1841	1841-11-01
8274	3.48	2.168	Madrid	Spain	40.99N	4.26W	139506	1841	1841-12-01
8275	0.111	2.523	Madrid	Spain	40.99N	4.26W	139507	1842	1842-01-01
8276	4.481	2.4	Madrid	Spain	40.99N	4.26W	139508	1842	1842-02-01
8277	8.495	2.035	Madrid	Spain	40.99N	4.26W	139509	1842	1842-03-01
8278	7.572	2.646	Madrid	Spain	40.99N	4.26W	139510	1842	1842-04-01
8279	14.323	1.793	Madrid	Spain	40.99N	4.26W	139511	1842	1842-05-01
8280	20.86	1.99	Madrid	Spain	40.99N	4.26W	139512	1842	1842-06-01
8281	22.638	2.478	Madrid	Spain	40.99N	4.26W	139513	1842	1842-07-01

Search - Ozone ¶

In [25]:
```
def mapsearchOzone(limit, ozo):
    parts = getPartitions("Ozone")
    ran = len(parts)
    return ozo[ozo['Hole Area'] >= int(limit)]

def reduceOzone(limit):
    parts = getPartitions("Ozone")
    outputs = []
    for i in range (1, len(parts.split(','))+1):
        ozo = readPartitions("Ozone" , i)
        outputs.append(mapsearchOzone(limit, ozo))
    r1=parts
    r2=ozo
    r3=outputs[-1]
    r4=pd.concat(outputs)
    return r1,r2,r3,r4
```



	Year	Hole Area	Minimum Ozone
0	2008	25.2	114
1	2009	22	107.9
2	2010	19.4	128.5
3	2011	24.7	106.5
4	2012	17.8	139.3
5	2013	21	132.7
6	2014	20.9	128.6
7	2015	25.6	117.2
8	2016	20.7	123.2
9	2017	17.4	141.8
10	2018	22.9	111.8
11	2019	9.2	167
13	2021	23.3	103.3



	Year	Hole Area	Minimum Ozone
0	2008	25.2	114
3	2011	24.7	106.5
7	2015	25.6	117.2

Above Reduced Partitions are merged and we get the final result:

	Year	Hole Area	Minimum Ozone
0	1993	24.2	112.6
1	1998	25.9	98.8
2	2000	24.8	98.7
3	2001	25	100.9
4	2003	25.8	108.7

Search - Sea

In [23]:
```
def mapsearchSea(start_year, end_year, sea):
    return(pd.merge(sea[sea['year'] <= end_year], sea[sea['year'] >= start_year], how ='inner', on =['year', 'total']))

def reduceSea(start_year, end_year):
    parts = getPartitions("Seas")
    outputs = []
    for i in range (1, len(parts.split(','))+1):
        sea = readPartitions("Seas", i).drop('Unnamed: 0', axis = 1)
        outputs.append(mapsearchSea(start_year, end_year,sea))
    return parts,sea,outputs[-1],pd.concat(outputs)
```

Explanation:
Getting the Partitions

['https://usctrojan-18644-default-rtdb.firebaseio.com/datanode/1/Seas1.json', 'https://usctrojan-18644-default-rtdb.firebaseio.com/datanode/2/Seas2.json', 'https://usctrojan-18644-default-rtdb.firebaseio.com/datanode/3/Seas3.json', 'https://usctrojan-18644-default-rtdb.firebaseio.com/datanode/4/Seas4.json', 'https://usctrojan-18644-default-rtdb.firebaseio.com/datanode/5/Seas5.json']

Sample Partition (last partition displayed but all Partitions from above will be used)

	total	year
1128	43.78	2016
1129	46.83	2016
1130	38.43	2016
1131	45.93	2016
1132	41.71	2016
1133	86.64	2016
1134	86.21	2016
1135	55.73	2016
1136	50	2016
1137	66.83	2016

Sample Reduced Partition (last reduced partition displayed but every partition will go through mapping function and we get the reduced partition)

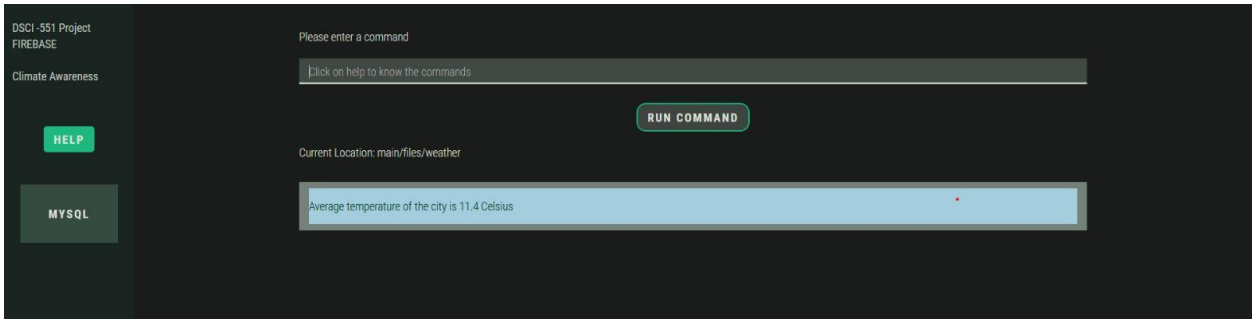
	total	year
0	43.78	2016
1	46.83	2016
2	38.43	2016
3	45.93	2016
4	41.71	2016
5	86.64	2016
6	86.21	2016
7	55.73	2016
8	50	2016
9	66.83	2016
10	67.22	2016
11	57.81	2016
12	56.97	2016
13	27.75	2016

Above Reduced Partitions are merged and we get the final result:

	total	year
0	105.07	1997
1	98.61	1997
2	79.49	1997
3	-4.33	1997
4	-20.25	1997
5	-24.07	1997
6	-18.47	1997
7	-15.66	1997
8	-61.44	1997
9	-61.9	1997
10	-56.3	1997
11	11.77	1997
12	37.74	1997

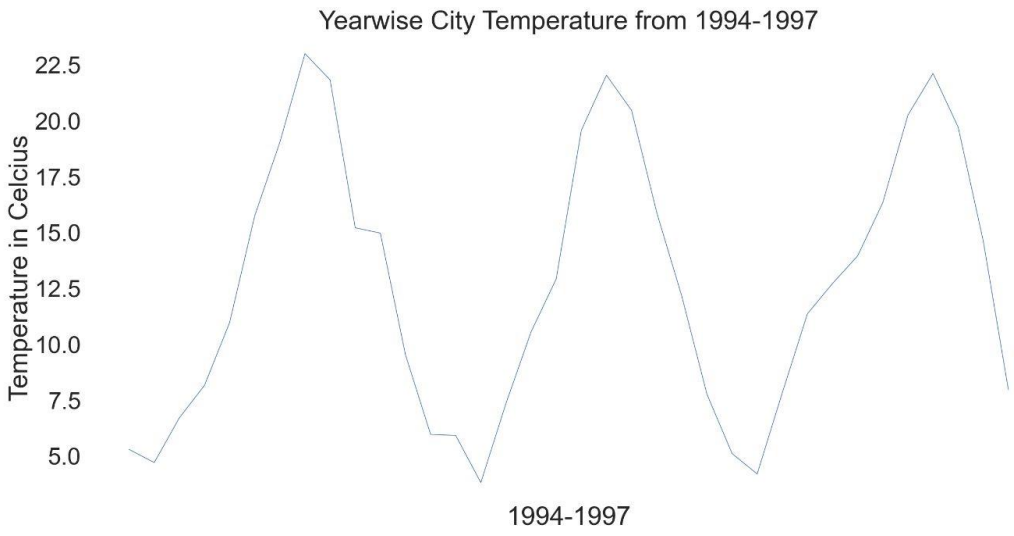
Analytics - Average Temperature

In [19]:
```
def CityTemp(city, glt):
    citytemps = glt[glt['City'] == city]['AverageTemperature'].dropna()
    return np.average(citytemps)

def avgtemp(city):
    parts = getPartitions("citytemp")
    ran = len(parts)
    result = []
    for i in range(1, len(parts.split(','))+1):
        glt = readPartitions("citytemp", i)
        if city in glt['City'].tolist():
            result.append(CityTemp(city, glt))
    return('Average temperature of the city is '+str(round(np.mean(result),2))+' Celsius')
```



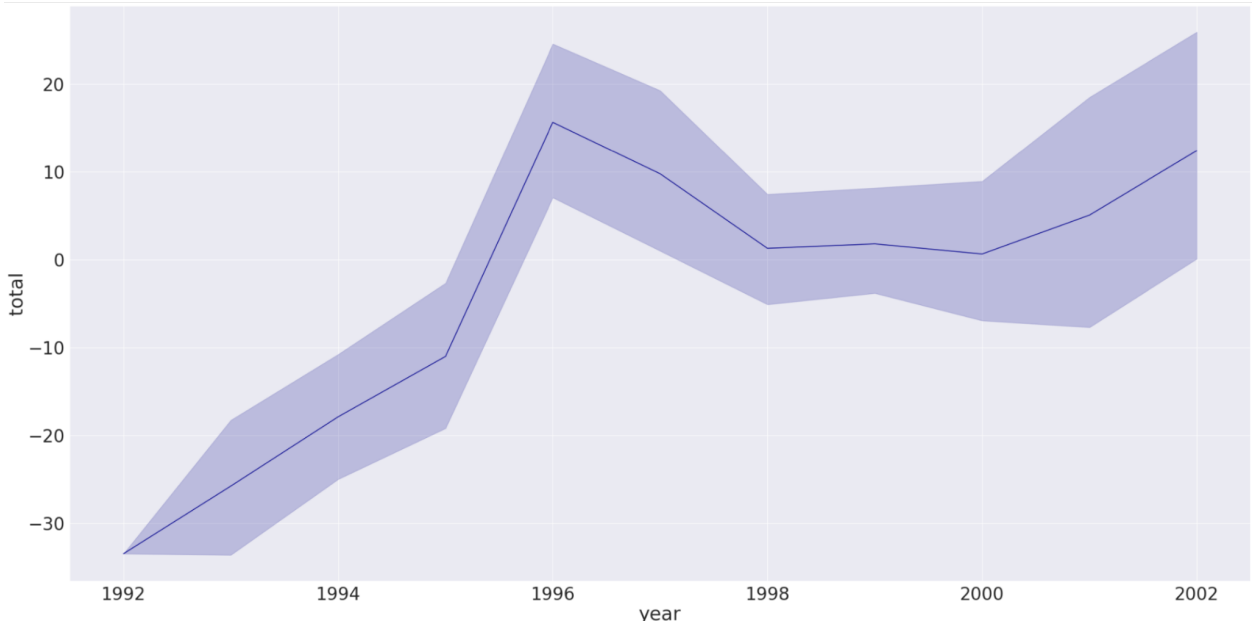
Analytics - City Graph

In [20]:
```
def citygraph(location,city, start_year, end_year):
    parts = getPartitions("CityTemp")
    ran = len(parts)
    index=0
    for i in range(1, len(parts.split(','))+1):
        glt = readPartitions("CityTemp" , i)
        glt = glt.reset_index().drop('index', axis = 1)
        for j in range (0, len(glt)):
            if glt['City'][j] == city:
                if glt['Years'][j] == start_year:
                    si = index
                if glt['Years'][j] == end_year:
                    ei = index
            index=index+1
    df = cat("CityTemp",location)
    sns.set(rc={'figure.figsize':(40,20)})
    sns.set(font_scale=5)
    ax = sns.lineplot(data= df["AverageTemperature"][si:ei])
    ax.set_xticks([])
    ax.set(xlabel=str(start_year) + '-' + str(end_year),
           ylabel='Temperature in Celcius',
           title='Yearwise City Temperature from ' + str(start_year) + '-' + str(end_year))
    return anvil.mpl_util.plot_image()
```



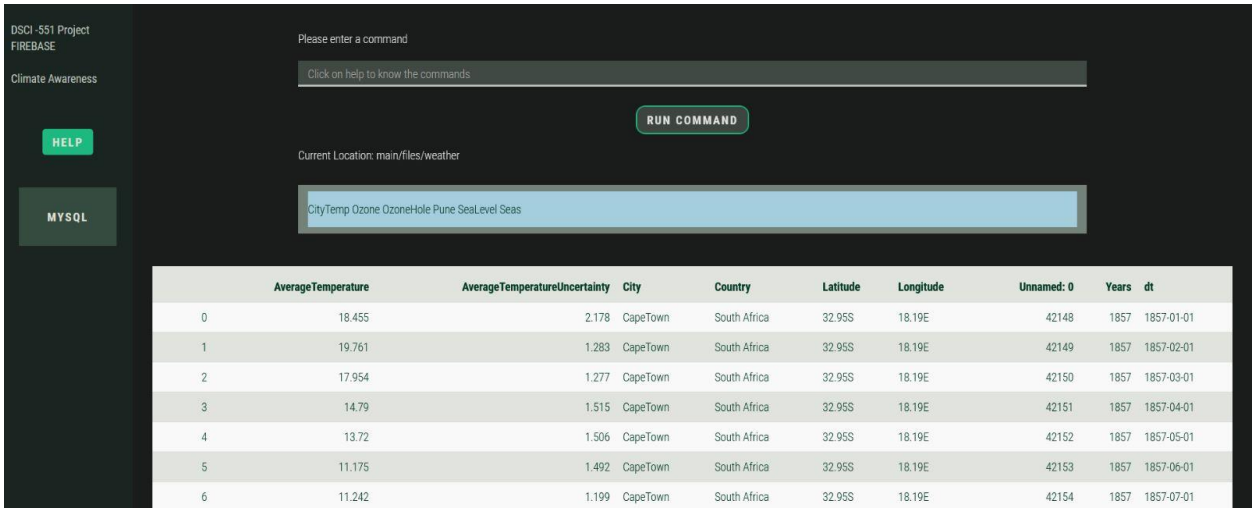
Analytics - Sea Levels

In [22]:
```
def maxyearsea(location,start_year, end_year):
    parts = getPartitions("Seas")
    ran = len(parts)
    temps = []
    index = 0
    for i in range(1, len(parts.split(','))+1):
        sea = readPartitions("Seas", i).reset_index()
        for s in range (0, len(sea)):
            if int(start_year) == int(sea['year'][s]):
                si = index
            if int(end_year) == int(sea['year'][s]):
                ei = index
            index = index+1
    temu = cat("Seas",location)
    sns.set(rc={'figure.figsize':(40,20)})
    sns.set(font_scale=5)
    ax=sns.lineplot(x = temu['year'][si:ei] , y = temu['total'][si:ei],color='darkblue')
    ax.set(ylabel='Sea Level',
           title='Yearwise Sea Levels ' + str(start_year) + '-' + str(end_year))
    return anvil.mpl_util.plot_image()
```

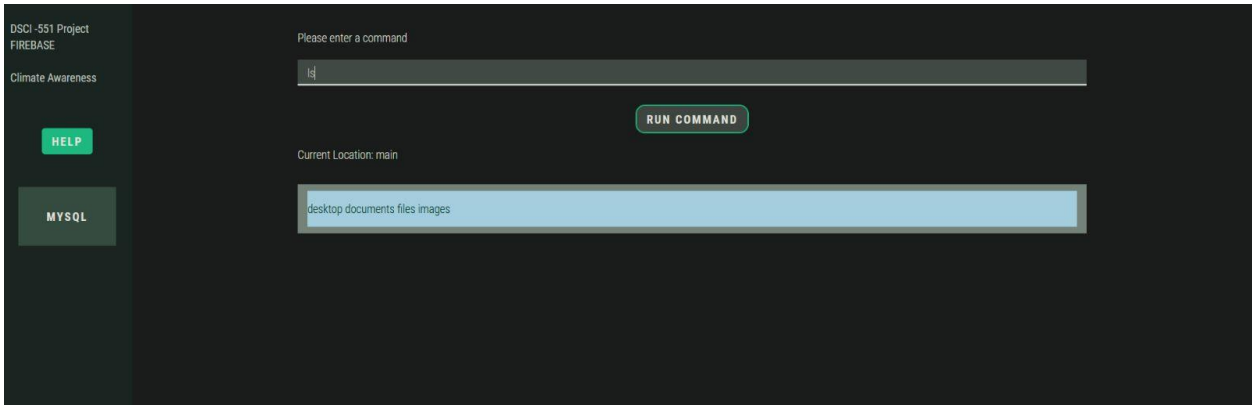


UI IMPLEMENTATION

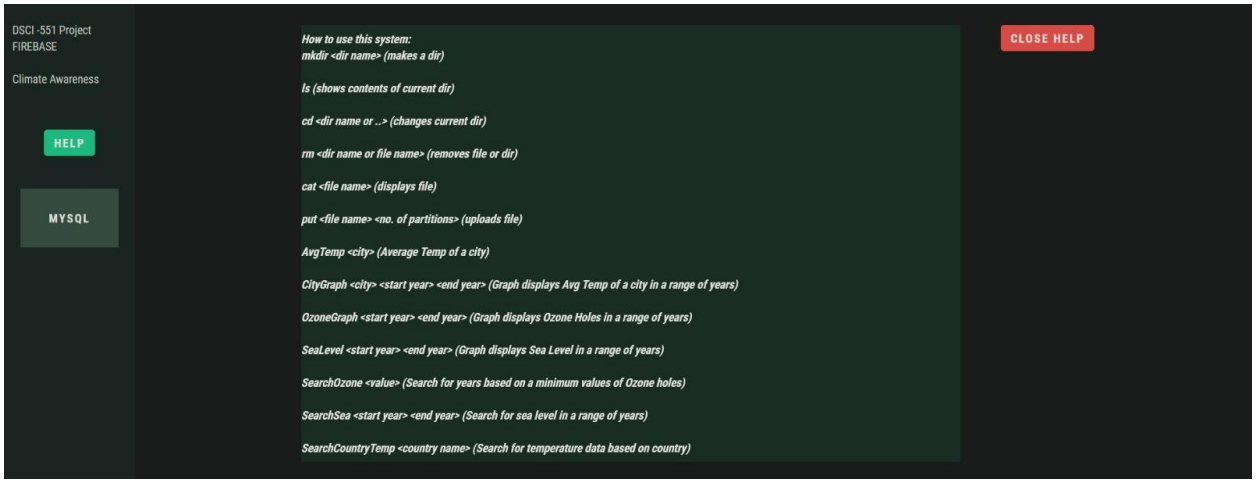
The UI has been implemented using Anvil Library using an uplink between server code and client code.
In our case, the Jupyter Notebooks are the server codes and the Anvil Front end is the client code.
Following are a few sample snippets of the User Interface created.



	AverageTemperature	AverageTemperatureUncertainty	City	Country	Latitude	Longitude	Unnamed: 0	Years	dt
0	18.455	2.178	CapeTown	South Africa	32.95S	18.19E	42148	1857	1857-01-01
1	19.761	1.283	CapeTown	South Africa	32.95S	18.19E	42149	1857	1857-02-01
2	17.954	1.277	CapeTown	South Africa	32.95S	18.19E	42150	1857	1857-03-01
3	14.79	1.515	CapeTown	South Africa	32.95S	18.19E	42151	1857	1857-04-01
4	13.72	1.506	CapeTown	South Africa	32.95S	18.19E	42152	1857	1857-05-01
5	11.175	1.492	CapeTown	South Africa	32.95S	18.19E	42153	1857	1857-06-01
6	11.242	1.199	CapeTown	South Africa	32.95S	18.19E	42154	1857	1857-07-01



To make the UI more intuitive and user friendly, a help box has been provided with the command syntaxes and input samples.



LEARNING EXPERIENCE:

1) Learnt the internal functioning of HDFS systems
2) Strengthened understanding of Firebase and MySQL tools
3) Learnt to integrate Jupyter Notebooks with an interactive User Interface
4) Understood inner workings of map reduce and also performed various functions.

HOW TO RUN OUR PROJECT

1) Run both Jupyter Notebooks.
2) Make sure that the csv files we will upload are present in the same directory as the Jupyter Notebooks.
3) Go to https://dsciproject.anvil.app and run.

DRIVE LINK TO VIDEO AND CODE:

https://drive.google.com/drive/folders/1iA3j730JAjmwHHEXTQOlDH8_fTjjzIo_?usp=sharing

---

TOOLS AND SOFTWARES:

Framework: Firebase, MySQL, Anvil

Language: Python

Library: JSON, Requests, Matplotlib, Pandas, NumPy, Seaborn.

We hope this project makes the right use of Statistics and Data Management tools and brings the fact that climate change is real and has grave consequences to light.