**Name**: U.Harsha

**College code**: 9530

**College name**: St. Mother Theresa Engineering College

**Team ID**:

**Naan Mudhalvan ID**: au953021104016

**Project Name**: Big Data Analysis with IBM cloud databases

**INTRODUCTION:**

Climate change is a pressing global issue, and big data analysis plays a crucial role in understanding its complexities. Big data encompasses vast sets of climate-related information, such as temperature records, satellite imagery, and greenhouse gas emissions data. By analyzing this data, scientists can identify trends, model future climate scenarios, and make informed policy decisions.

Big data analytics enables researchers to:

1. Detect patterns and trends
2. Model climate scenarios
3. Monitor environmental changes
4. Assess climate impacts
5. Track greenhouse gas emissions

In summary, big data analysis is a powerful tool in our efforts to address and mitigate the impacts of climate change by providing valuable insights and supporting informed decision-making.

**PROBLEM SOLUTION:**

To find a solution to your problem statement, you can use IBM Cloud Databases to store and manage vast datasets. You can then use the built-in analytics tools to uncover hidden insights from the data. Once you have identified the insights, you can visualize them using the built-in visualization tools. This will help you derive valuable business intelligence from the data and make informed decisions. Using big data to address climate change involves collecting, analyzing, and applying large volumes of data to develop informed strategies and solutions. Here's how it can be done:

1. Data Collection: Collect climate-related data from various sources, such as satellites, weather stations, IoT sensors, and social media. Include data on temperature, carbon emissions, air quality, deforestation, and more.

2. Data Analysis: Employ machine learning algorithms to process and analyze the data to identify trends, correlations, and anomalies. Predict future climate patterns, extreme weather events, and their impacts.

3. Energy Efficiency: Use data to optimize energy consumption in industries and building. Implement smart grids and sensors to monitor and control energy usage in real-time.

4. Carbon Footprint Reduction: Track carbon emissions across industries and transportation. Identify areas for emission reductions and prioritize actions based on data insights.

5. Renewable Energy: Analyze weather and energy production data to optimize renewable energy sources like solar and wind. Predict energy generation and demand to ensure a stable grid.

6. Climate Resilience: Use data to assess vulnerabilities and develop adaptive strategies for communities and infrastructure. Monitor changes in sea levels, temperature, and extreme events to enhance preparedness.

7. Agriculture and Land Use: Optimize farming practices using data on weather, soil conditions, and crop performance. Promote reforestation and sustainable land management based on data-driven insights.

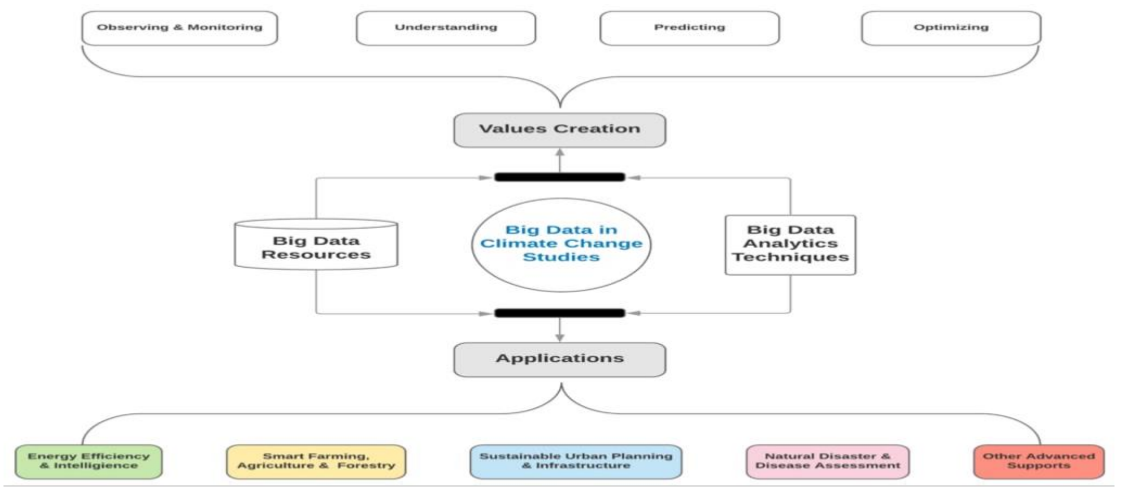
8. Behavioral Change: Leverage data to influence consumer behavior through personalized recommendations and incentives. Encourage eco-friendly choices in transportation, consumption, and energy use.

9. Policy and Advocacy: Provide policymakers with data-driven evidence to support climate policies. Advocate for informed decisions and international cooperation based on shared data.

10. Monitoring and Reporting: Continuously monitor progress toward climate goals using real-time data. Share transparent reports to hold governments and industries accountable.

Big data analytics can empower governments, businesses, and individuals to make informed decisions, reduce greenhouse gas emissions, and adapt to the changing climate. It's a powerful tool in the fight against climate change.

**METHODOLOGY:**


Observing & Monitoring
Understanding
Predicting
Optimizing
Values Creation
Big Data Resources
Big Data in Climate Change Studies
Big Data Analytics Techniques
Applications
Energy Efficiency & Intelligience
Smart Farming, Agriculture & Forestry
Sustainable Urban Planning & Infrastructure
Natural Disaster & Disease Assessment
Other Advanced Supports

**EXPLANATION:**

Big data analytics plays a crucial role in climate change studies by helping scientists and researchers better understand the complex and interconnected factors influencing climate change.

It consists of two major components. They are

1. Big Data Resources
2. Big Data Analytics Techniques

**Big Data Resources:**

"Big Data Resources" typically refers to various types of information, tools, and assets that are valuable for working with big data. Big data refers to the vast volume, velocity, and variety of data that organizations and researchers deal with, often requiring specialized resources to manage, analyze, and derive insights from this data.

**Big Data Analytics Techniques:**

Big data analytics techniques encompass a wide range of methods and approaches used to analyze and extract insights from large and complex datasets. These techniques are essential for uncovering patterns, trends, correlations, and valuable information within massive volumes of data.

These two components can be access for **Value Creation** and **Applications.**

**Value Creation:**

Value creation refers to the process of generating additional worth or benefits for stakeholders through various activities, products, services, or investments. It is a fundamental concept in business, economics, and other fields, and it plays a central role in understanding how organizations and individuals contribute to economic, social, and environmental well-being. Value creation can be divided into four parts.

a. Observing and Monitoring
b. Understanding
c. Predicting
d. Optimizing

**Applications:**

Applications in big data analytics are diverse and cover a wide range of industries and use cases. Big data analytics is used to extract valuable insights, make data-driven decisions, and solve complex problems in various domains. Applications can be divided into five parts.

a. Energy Efficiency and Intelligence
b. Smart Farming, Agriculture and Forestry
c. Sustainable Urban Planning and Infrastructure
d. Natural Disaster & Disaster Assessment

## e. Other Advanced Supports

**OBJECTIVES:**

- Provision a SQL database
- Create the database schema (table) and load data
- Deploy a pre-built containerized app to Code Engine
- Connect the app and database service (share credentials)
- Monitor, Secure, Backup & Recovery of cloud databases

### Step 1: Provision the SQL Database

Start by creating an instance of the Db2 Warehouse on Cloud service.

1. Visit the IBM Cloud® console. Click on **Catalog** in the top navigation bar.
2. Click on **Databases** on the left pane and select **Db2 Warehouse**.
3. Pick the **Flex One** plan and change the suggested service name to **sqldatabase** (you will use that name later on). Pick a resource group and a location for the deployment of the database.
4. Click on **Create**. The provisioning starts.
5. In the **Resource List**, locate the new instance under **Databases** and wait for it to be available (sometimes you may need to refresh the page). Click on the entry for your Db2 Warehouse on Cloud service.
6. Click on **Open Console** to launch the database console.

### Step 2: Create a table

1. In the console for Db2 Warehouse on Cloud click on the upper left menu icon, then **Run SQL** in the navigation bar.
2. Click on the + symbol (**Add a new script**) next to the **Untitled - 1** tab.
3. Click on **From file** and select the file cityschema.txt from the GitHub repository that was previously cloned to your local directory and open it.
4. Click on **Run all** to execute the statement. It should show a success message.

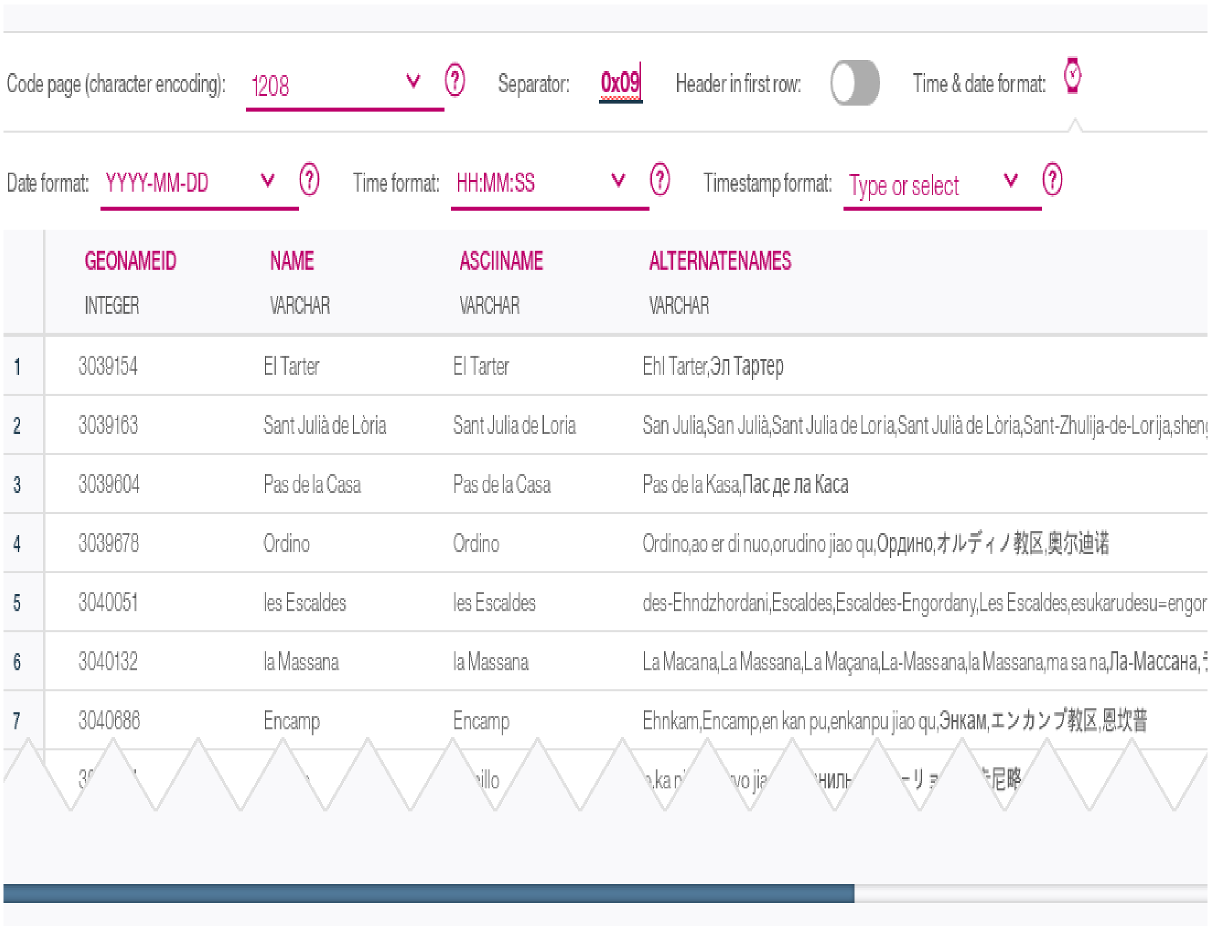
### Step 3: Load data

Now that the table "cities" has been created, you are going to load data into it. This can be done in different ways, for example from our local machine or from cloud object storage (COS) or Amazon S3 interface. We are going to upload data from our machine. During that process, you adapt the table structure and data format to fully match the file content.

1. In the console for Db2 Warehouse on Cloud click on the upper left menu icon, then **Data** in the navigation bar.
2. As **Source** keep the selection on **My Computer**.
3. Under **File selection**, click on **Drag a file here or browse files** to locate and pick the file "cities1000.txt" you downloaded in the first section of this guide.
4. Click **Next** to get to the **Target** overview with a **Schema** selection. Choose the schema **BLUADMIN**, then the table **CITIES**. Click on **Next** again.

Because the table is empty it does not make a difference to either append to or overwrite existing data.

5. Now customize how the data from the file "cities1000.txt" is interpreted during the load process. First, disable **Header in first row** because the file contains data only.
6. Next, type in **0x09** as separator. It means that values within the file are delimited by tab(ulator).
7. Last, pick "YYYY-MM-DD" as date format. Now, everything should look similar to what is shown in this screen capture.



Code page (character encoding): 1208 | Separator: 0x09 | Header in first row: off | Time & date format:

Date format: YYYY-MM-DD | Time format: HH:MM:SS | Timestamp format: Type or select

| | GEONAMEID<br>INTEGER | NAME<br>VARCHAR | ASCIINAME<br>VARCHAR | ALTERNATENAMES<br>VARCHAR |
|---|---|---|---|---|
| 1 | 3039154 | El Tarter | El Tarter | Ehl Tarter,Эл Тартер |
| 2 | 3039163 | Sant Julià de Lòria | Sant Julia de Loria | San Julia,San Julià,Sant Julia de Loria,Sant Julià de Lòria,Sant-Zhulija-de-Lorija,shen |
| 3 | 3039604 | Pas de la Casa | Pas de la Casa | Pas de la Kasa,Пас де ла Каса |
| 4 | 3039678 | Ordino | Ordino | Ordino,ao er di nuo,orudino jiao qu,Ордино,オルディノ教区,奥尔迪诺 |
| 5 | 3040051 | les Escaldes | les Escaldes | des-Ehndzhordani,Escaldes,Escaldes-Engordany,Les Escaldes,esukarudesu=engor |
| 6 | 3040132 | la Massana | la Massana | La Macana,La Massana,La Maçana,La-Massana,la Massana,ma sa na,Ла-Массана, |
| 7 | 3040686 | Encamp | Encamp | Ehnkam,Encamp,en kan pu,enkanpu jiao qu,Энкам,エンカンプ教区,恩坎普 |

Screen capture showing the sampled data

8. Click **Next** and you are offered to review the load settings. Agree and click **Begin Load** to start loading the data into the **CITIES** table. The progress is displayed. Once the data is uploaded it should only take few seconds until the load is finished and some statistics are presented.
9. Click on **View Table** to browse the data. You may scroll down or click on column names to change the sort order.

**Step 4: Verify Loaded Data Using SQL**

The data has been loaded into the relational database. There were no errors, but you should run some quick tests anyway. Use the built-in SQL editor to type in and execute some SQL statements.

1. In the left navigation click on **Run SQL** to get back to the SQL editor. Click on the + symbol (**Add new script**) and **Create new** to create a new editor tab.

   Instead of the built-in SQL editor you can use cloud-based and traditional SQL tools on your desktop or server machine with Db2 Warehouse on Cloud. The connection information can be found in the **Administration** menu in the left navigation.

2. In the editor type or copy in the following query:
3. `select count(*) from cities;`

   Select the text of the query, then, in dropdown next to **Run All**, choose **Run selected**. In the section with results, the same number of rows as reported by the load process should be shown.

4. In the "SQL Editor" enter the following statement on a new line:
5. `select countrycode, count(name) from cities`
6. `group by countrycode`
7. `order by 2 desc;`

   Mark the text of the above statement and click the **Run selected** button. Only this statement is executed, returning some by country statistics in the results section.

8. Finally, run the following statement similarly to retrieve details about San Francisco in California:
9. select*from cities
10. where name='San Francisco'
11. and countrycode='US';

**Step 5: Deploy the application code**

Change back to the terminal. Now you are going to deploy the application code, using a pre-built container image. You can modify the application code and build the container image on your own. See the instructions in the GitHub repository for details.

1. If you are not logged in, use `ibmcloud login` or `ibmcloud login --sso` to log in interactively. Set the region and resource group to where the database has been provisioned. Replace **RESOURCE_GROUP** and **REGION** accordingly.
2. ibmcloud target -g RESOURCE_GROUP -r REGION
3. Create a new Code Engine project named **sqldatabase**:
4. ibmcloud ce project create --name sqldatabase

   Select the new project as the active one:

   ibmcloud ce project select --name sqldatabase

5. Then, deploy the app naming it **worldcities**.
6. ibmcloud ce app create

7. Last, create a service binding between the existing Db2 Warehouse on Cloud database and the app:
8. ibmcloud ce application bind --name worldcities --service-instance sqldatabase

   Once the binding is created, a new app revision is started.

9. Now you can check the app details for its status and to retrieve its URL:
10. ibmcloud ce app get --name worldcities

**Step 6: Security, Backup & Recovery, Monitoring**

The Db2 Warehouse on Cloud is a managed service. IBM takes care of securing the environment, daily backups and system monitoring. When you are using one of the enterprise plans there are several options to manage access and to configure enhanced data encryption.

In addition to the traditional administration options the Db2 Warehouse on Cloud service also offers a REST API for monitoring, user management, utilities, load, storage access and more.

**Step 7: Test the App**

The app to display city information based on the loaded data set is reduced to a minimum .It offers a search form to specify a city name - names are case sensitive - and few preconfigured cities. They are translated to either `/search?name=cityname` (search form) or `/city/cityname` (directly specified cities). Both requests are served from the same lines of code in the background. The `cityname` is passed as value to a prepared SQL statement using a parameter marker for security reasons. The rows are fetched from the database and passed to an HTML template for rendering.

**Step 8: Cleanup**

To clean up resources follow these steps:

1. Visit the IBM Cloud® Resource List.
2. In the Code Engine section locate the project **sqldatabase**. Click on the three dots and select **Delete** to delete the project and its app.
3. Locate the database `sqldatabase` under **Databases**. Again, click on the three dots and select **Delete** to delete the database.

   Depending on the resource it might not be deleted immediately, but retained (by default for 7 days). You can reclaim the resource by deleting it permanently or restore it within the retention period.

**SOURCE CODE:**

```
In [1]:

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data_country = pd.read_csv("../input/climate-change-earth-surface-temperature-data/GlobalLandTemperaturesByCountry.csv")
data_greece = data_country[data_country["Country"] =="Greece"].copy()
```

```
data_greece["dt"] = pd.to_datetime(data_greece["dt"])
data_global  =  pd.read_csv("../input/climate-change-earth-surface-temperature-data/GlobalTe
mperatures.csv")
data_global["dt"] = pd.to_datetime(data_global["dt"])
co2_ppm = pd.read_csv("../input/carbon-dioxide/archive.csv")
```

In [2]:

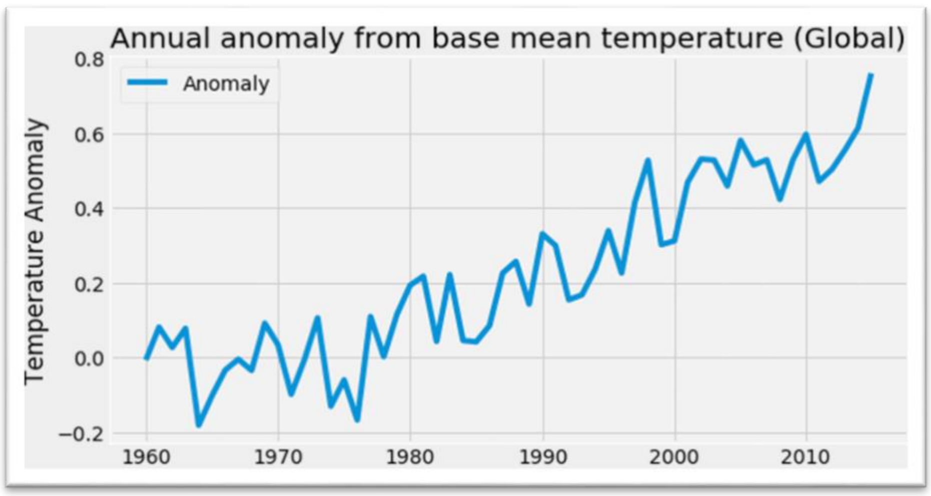
```
annual_mean_global=data_global.groupby(data_global["dt"].dt.year).mean()
reference_temperature_global=annual_mean_global.loc[1951:1980].mean()["LandAndOcean
AverageTemperature"]
annual_mean_global["Anomaly"]=annual_mean_global["LandAndOceanAverageTemperatur
e"]-reference_temperature_global
annual_mean_greece=data_greece.groupby(data_greece["dt"].dt.year).mean()
reference_temperature_greece=annual_mean_greece.loc[1951:1980].mean()["AverageTempe
rature"]
annual_mean_greece["Anomaly"]=annual_mean_greece["AverageTemperature"]-reference_t
emperature_greece
```

I calculated the mean temperature of the 1951 - 1980 period to establish the global base mean temperature. This is standard practice in climate science. The deviation from this temperature is added in the Anomaly column. I also created a new dataframe for my home country (Greece), and repeated this process.

In [3]:

```
plt.figure()
plt.style.use("fivethirtyeight")
annual_mean_global.loc[1960:2015]["Anomaly"].plot(figsize=(10,5),grid=True,legend=True
)
plt.title("Annual anomaly from base mean temperature (Global)")
plt.xlabel('')
plt.ylabel('Temperature Anomaly')
plt.show()
```

Out [3]:

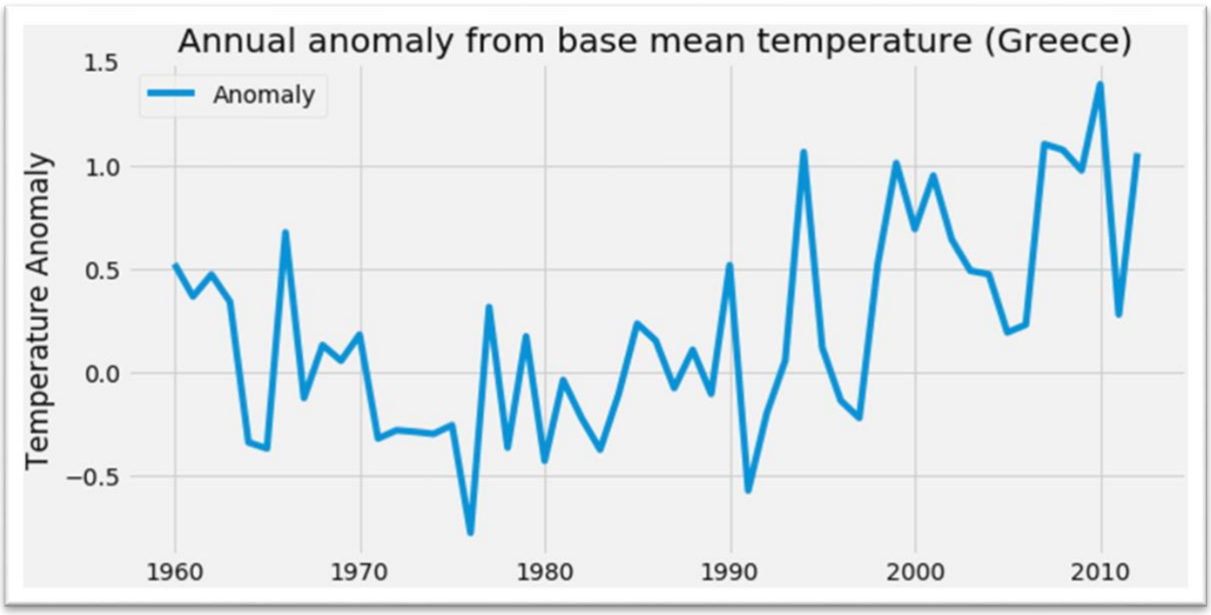


As we can see, the global mean temperature has grown steadily the past decades, leading to a temperature anomaly of about 0.75 celsius in 2015. As expected, this result is consistent with the scientific consensus on climate change.

In [4]:

```
plt.figure()
plt.style.use("fivethirtyeight")
annual_mean_greece.loc[1960:2012]["Anomaly"].plot(figsize = (10,5), grid=True, legend=Tr
ue)
plt.title("Annual anomaly from base mean temperature (Greece)")
plt.xlabel('')
plt.ylabel('Temperature Anomaly')
plt.show()
```

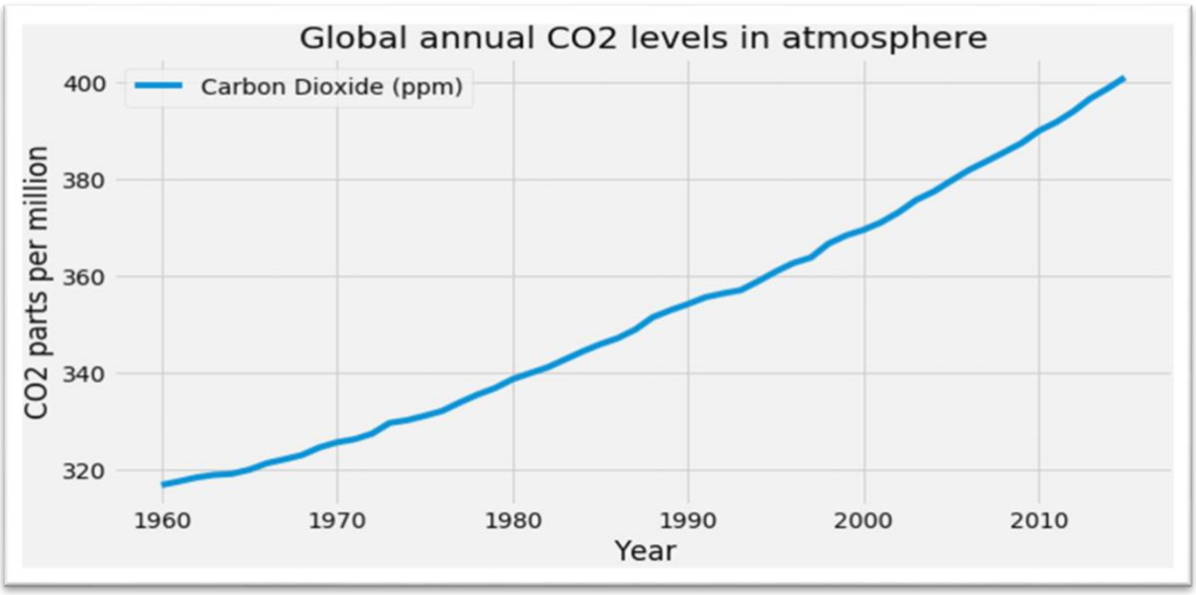
Out [4]:



The temperature has also steadily increased in my home country Greece.

In [5]:

```
plt.figure()
plt.style.use("fivethirtyeight")
annual_co2_ppm = co2_ppm.groupby(co2_ppm["Year"]).mean()
annual_co2_ppm.loc[1960:2015]["Carbon Dioxide (ppm)"].plot(figsize = (10,5), grid=True, l
egend=True)
plt.title("Global annual CO2 levels in atmosphere")
plt.ylabel("CO2 parts per million")
plt.show()
```



The CO2 levels in atmosphere have steadily risen in the 1950-2010 period, indicating a linear relation between greenhouse gases and global temperature.

In [6]:

```
annual_co2_temp = pd.merge(annual_mean_global.loc[1960:2015], annual_co2_ppm.loc[196
0:2015], left_index=True, right_index=True)
annual_co2_temp = annual_co2_temp[["LandAndOceanAverageTemperature", "Anomaly", "
Carbon Dioxide (ppm)"]].copy()
annual_co2_temp.corr()
```

Out [6]:

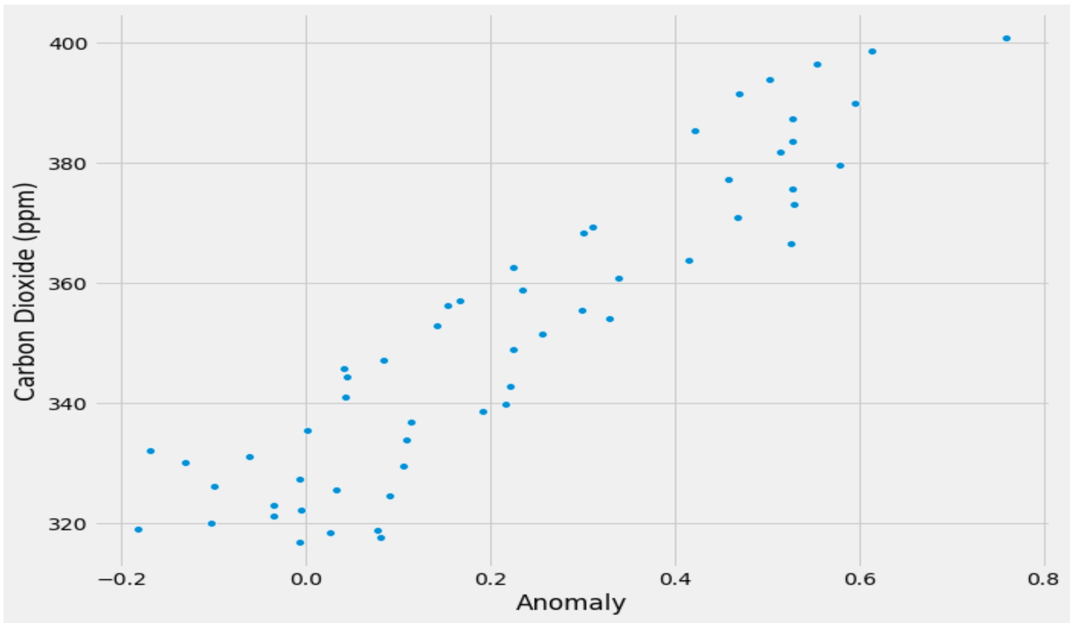
| | Land And Ocean Average Temperature | Anomaly | Carbon Dioxide (ppm) |
|---|---|---|---|
| Land And Ocean Average Temperature | 1.000000 | 1.000000 | 0.923603 |
| Anomaly | 1.000000 | 1.000000 | 0.923603 |
| Carbon Dioxide (ppm) | 0.923603 | 0.923603 | 1.000000 |

The correlation coefficient of CO2 and temperature anomaly is 0.92, confirming the linear relation between the two variables.

In [7]:
```
plt.figure(figsize=(10,8))
sns.scatterplot(x="Anomaly",y="Carbon Dioxide (ppm)", data=annual_co2_temp)
```

Out[7]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4819909d30>
```



This scatter plot visualizes the linear relation between CO2 levels and temperature anomaly.

**CONCLUSION:**

Finally, this big data-driven climate change research has provided vital insights into the ongoing environmental difficulties we face. We've been able to analyse and anticipate climate trends, follow the impact of greenhouse gas emissions, and design mitigation and adaptation plans by leveraging massive databases. The use of big data has improved our ability to make educated decisions and implement successful climate change policy. However, in order to solve this vital issue and strive towards a more sustainable future, we must continue this research, enhance our models, and collaborate across disciplines.

**GitHub Link:**

**Faculty Evaluator Mail ID:**

danipackiaseeli@mtec.ac.in