ASSIGNMENT NO: 1

Title: Single node/Multiple node Hadoop Installation.

Learning Objectives:

Problem Statement: Study and perform Hadoop Installation on a)Single Node b)Multiple Node

Software and Hardware Requirement: PC with Ubuntu

Theory:

What is Hadoop:

Apache Hadoop is an open source framework that is used to efficiently store and process large datasets ranging in size from gigabytes to petabytes of data. Instead of using one large computer to store and process the data, Hadoop allows clustering multiple computers to analyze massive datasets in parallel more quickly.

Hadoop consists of four main modules:

- Hadoop Distributed File System (HDFS) – A distributed file system that runs on standard or low-end hardware. HDFS provides better data throughput than traditional file systems, in addition to high fault tolerance and native support of large datasets.
- Yet Another Resource Negotiator (YARN) – Manages and monitors cluster nodes and resource usage. It schedules jobs and tasks.
- MapReduce – A framework that helps programs do the parallel computation on data. The map task takes input data and converts it into a dataset that can be computed in key value pairs. The output of the map task is consumed by reduce tasks to aggregate output and provide the desired result.
- Hadoop Common – Provides common Java libraries that can be used across all modules.

Steps to install Hadoop on Ubuntu:

```
sudo apt-get update

sudo apt install openjdk-8-jdk

cd /usr/lib/jvm

ls

cd

sudo nano .bashrc

//add following lines at the end

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

export PATH=$PATH:/usr/lib/jvm/java-8-openjdk-amd64/bin
```

```
export HADOOP_HOME=~/hadoop-3.3.4/

export PATH=$PATH:$HADOOP_HOME/bin

export PATH=$PATH:$HADOOP_HOME/sbin

export HADOOP_MAPRED_HOME=$HADOOP_HOME

export YARN_HOME=$HADOOP_HOME

export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop

export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native

export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"

export HADOOP_STREAMING=$HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.4.jar

export HADOOP_LOG_DIR=$HADOOP_HOME/logs

export PDSH_RCMD_TYPE=ssh

//ctrl+o, enter, ctrl+x

sudo apt-get install ssh

tar -zxvf ~/Downloads/hadoop-3.3.4.tar.gz

cd hadoop-3.3.4/etc/hadoop

ls

sudo nano hadoop-env.sh

//uncomment JAVA_HOME

JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

(set the path for JAVA_HOME)

//ctrl+o, enter, ctrl+x

sudo nano core-site.xml

//add configuration at the end

<configuration>

 <property>

 <name>fs.defaultFS</name>

 <value>hdfs://localhost:9000</value>  </property>

 <property>
```

<name>hadoop.proxyuser.dataflair.groups</name> <value>*</value>

</property>

<property>

<name>hadoop.proxyuser.dataflair.hosts</name> <value>*</value>

</property>

<property>

<name>hadoop.proxyuser.server.hosts</name> <value>*</value>

</property>

<property>

<name>hadoop.proxyuser.server.groups</name> <value>*</value>

</property>

</configuration>

//ctrl+o, enter, ctrl+x

sudo nano hdfs-site.xml

//add configuration at the end

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

</configuration>

//ctrl+o, enter, ctrl+x

sudo nano mapred-site.xml

//add configuration at the end

<configuration>

<property>

<name>mapreduce.framework.name</name> <value>yarn</value>

</property>

<property>

<name>mapreduce.application.classpath</name>

<value>$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/*:$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/lib/*</value>

</property>

</configuration>

//ctrl+o, enter, ctrl+x

sudo nano yarn-site.xml

//add configuration at the end

<configuration>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.env-whitelist</name>

<value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH_PREP END_DISTCACHE,HADOOP_YARN_HOME,HADOOP_MAPRED_HOME</value>

</property>

</configuration>

//ctrl+o, enter, ctrl+x

ssh localhost

ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa

cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

chmod 0600 ~/.ssh/authorized_keys

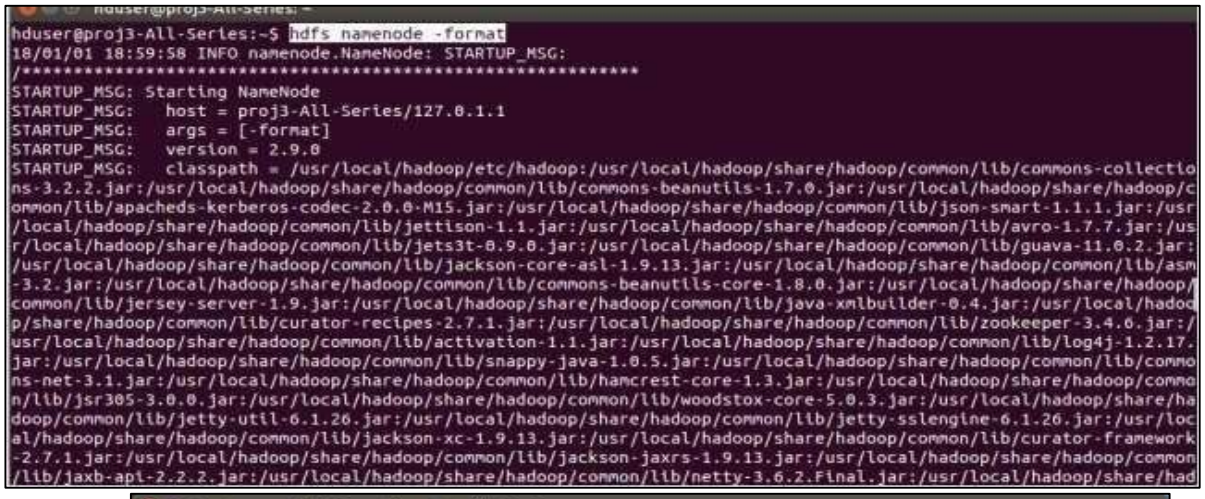
hadoop-3.3.4/bin/hdfs namenode -format

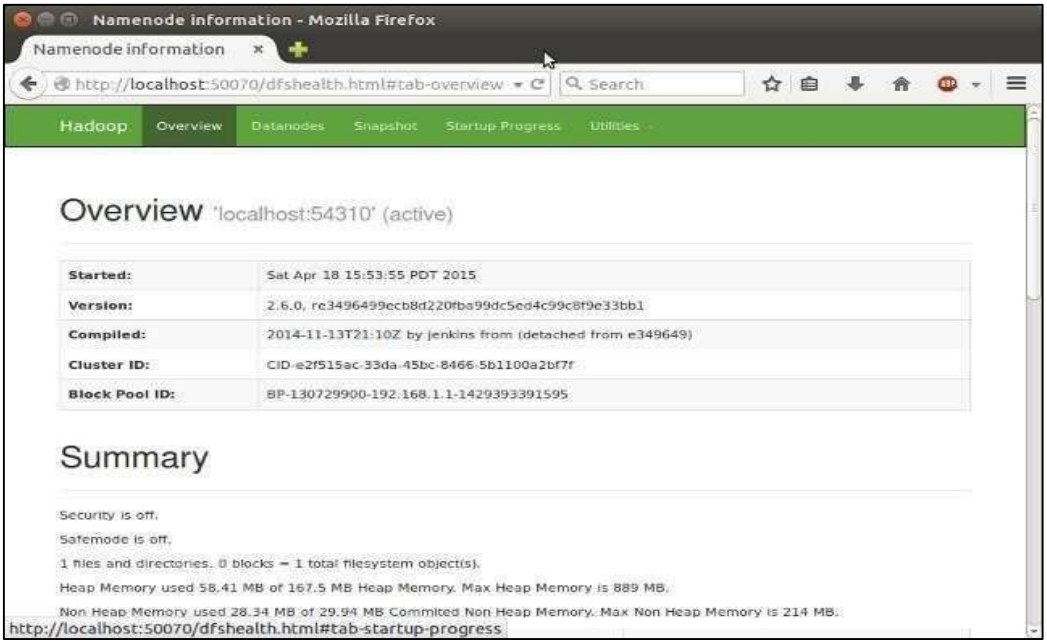
export PDSH_RCMD_TYPE=ssh

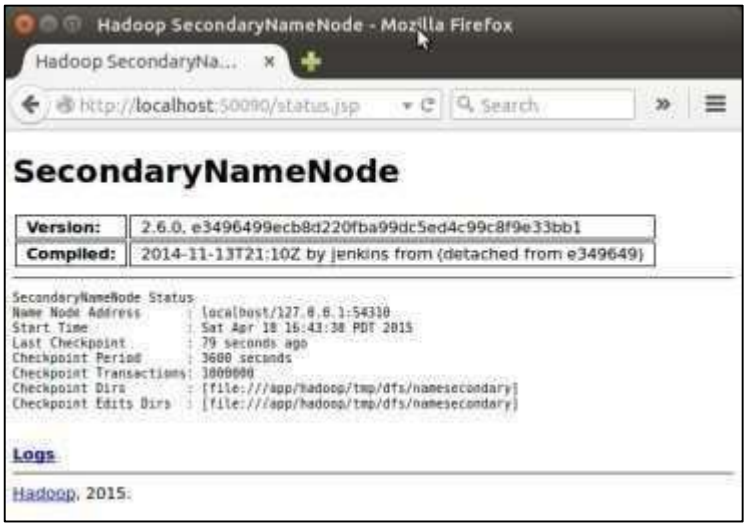
start-all.sh

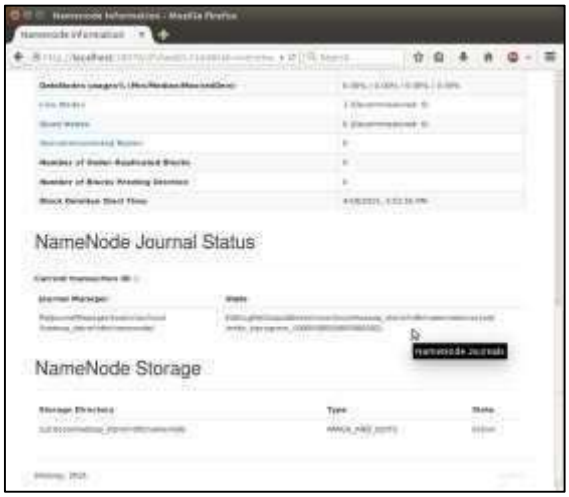
//open in browser localhost:9870

Ouptut:









Conclusion: Single node and multi-node installation of hadoop .

`

ASSIGNMENT NO: 2

Title: Design a distributed application using MapReduce(Using Java) which processes a log file of a system. List out the users who have logged for maximum period on the system. Use simple log file from the Internet and process it using a pseudo distribution mode on Hadoop platform.Learning Objectives:

Problem Statement: Study and perform program using map reduce on Hadoop.

Software and Hardware Requirement: Implement mapreduce on Ubuntu.

Theory:

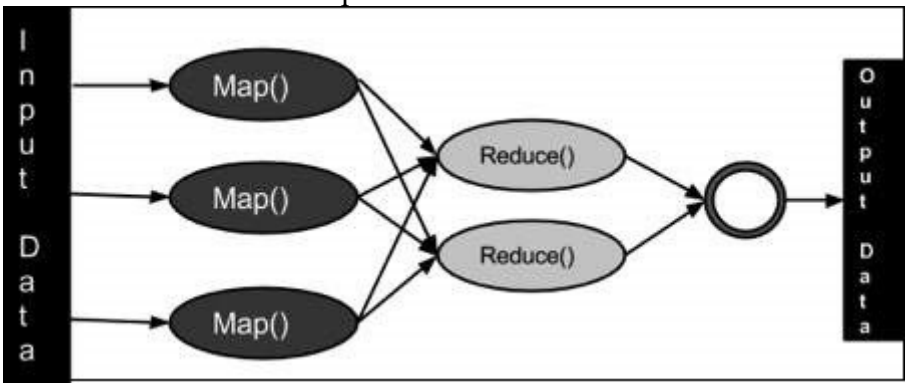
What is MapReduce?

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into *mappers* and *reducers* is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

The Algorithm

- Generally MapReduce paradigm is based on sending the computer to where the data resides!
- MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.
  - **Map stage** − The map or mapper's job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.
  - **Reduce stage** − This stage is the combination of the **Shuffle** stage and the **Reduce** stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.
- During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.
- The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.
- Most of the computing takes place on nodes with data on local disks that reduces the network traffic.
- After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.



Inputs and Outputs (Java Perspective)

The MapReduce framework operates on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types.

The key and the value classes should be in serialized manner by the framework and hence, need to implement the Writable interface. Additionally, the key classes have to implement the Writable-Comparable interface to facilitate sorting by the framework. Input and Output types of a **MapReduce job** − (Input) <k1, v1> → map → <k2, v2> → reduce → <k3, v3>(Output).

Implementation Steps:

1. Create folder Data on Desktop and copy all given files(i. access_log_short.txt – Data set to analyze

ii. SalesCountryDriver.java : Manages input and output from Mapper and Reducer

iii. SalesMapper.java : Reads input file

iv. SalesCountryReducer.java : Reads the input from mapper file)

2. Open file named access_log_short.txt using libreoffice/wps and use separator as "-" (Uncheck remaining options), save the file as text CSV format

3. su – hduser (Change the user)

4. Sudo mkdir analyzelogs (Create folder analyzelogs)

and copy the contents of Data folder into analyzelogs folder using command

5. sudo cp /home/ll01/Desktop/Data/* ~/analyzelogs

6. Give permissions using

sudo chmod -R 777 analyzelogs /

7. Change the owner to hduser using following

sudo chown -R hduser analyzelogs /

8. cd analyzelogs

9. sudo chmod +r *.*

10. export CLASSPATH="$HADOOP_HOME/share/hadoop/mapreduce/hadoop- mapreduce-clientcore-2.9.0.jar:$HADOOP_HOME/share/hadoop/mapreduce/hadoop- mapreduce-client-common2.9.0.jar:$HADOOP_HOME/share/hadoop/common/hadoop-common2.9.0.jar:~/analyzelogs/SalesCountry/*:$HADOOP_HOME/lib/*"

10. sudo gedit Manifest.txt

11. Add following line and press enter and save the file Main-Class: SalesCountry.SalesCountryDriver

12. javac -d . SalesMapper.java SalesCountryReducer.java SalesCountryDriver.java

13. jar -cfm analyzelogs.jar Manifest.txt SalesCountry/*.class
14. start-dfs.sh
15. Start-yarn.sh
16. jps
17. cd analyzelogs/
18. sudo mkdir ~/input
19. sudo cp access_log_short.csv ~/input/

Input:

1) Mapper file-

```
package SalesCountry; import java.io.IOException;

import org.apache.hadoop.io.IntWritable; import org.apache.hadoop.io.LongWritable; import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.*;

public class SalesMapper extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> { private final static IntWritable one = new IntWritable(1);


public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {

String valueString = value.toString();

String[] SingleCountryData = valueString.split("-"); output.collect(new Text(SingleCountryData[0]), one);

}

}
```

2) Reducer file:

```
package SalesCountry;

import java.io.IOException; import java.util.*;

import org.apache.hadoop.io.IntWritable; import org.apache.hadoop.io.Text; import org.apache.hadoop.mapred.*;
```

```
public class SalesCountryReducer extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {



public void reduce(Text t_key, Iterator<IntWritable> values, OutputCollector<Text,IntWritable> output, Reporter reporter) throws IOException {

Text key = t_key;

int frequencyForCountry = 0; while (values.hasNext()) {

// replace type of value with the actual type of our value IntWritable value = (IntWritable) values.next(); frequencyForCountry += value.get();



}

output.collect(key, new IntWritable(frequencyForCountry));



}

}
```

3) Driver File package SalesCountry;

```
import org.apache.hadoop.fs.Path; import org.apache.hadoop.io.*; import org.apache.hadoop.mapred.*;



public class SalesCountryDriver {

public static void main(String[] args) { JobClient my_client = new JobClient();

// Create a configuration object for the job

JobConf job_conf = new JobConf(SalesCountryDriver.class);



// Set a name of the Job job_conf.setJobName("SalePerCountry");

// Specify data type of output key and value job_conf.setOutputKeyClass(Text.class); job_conf.setOutputValueClass(IntWritable.class);



// Specify names of Mapper and Reducer Class job_conf.setMapperClass(SalesCountry.SalesMapper.class); job_conf.setReducerClass(SalesCountry.SalesCountryReducer.class);

// Specify formats of the data type of Input and output job_conf.setInputFormat(TextInputFormat.class); job_conf.setOutputFormat(TextOutputFormat.class);



// Set input and output directories using command line arguments,
```

//arg[0] = name of input directory on HDFS, and arg[1] = name of output directory to be created to store the output file.

FileInputFormat.setInputPaths(job_conf, new Path(args[0])); FileOutputFormat.setOutputPath(job_conf, new Path(args[1]));

my_client.setConf(job_conf); try {

// Run the job JobClient.runJob(job_conf);

} catch (Exception e) {

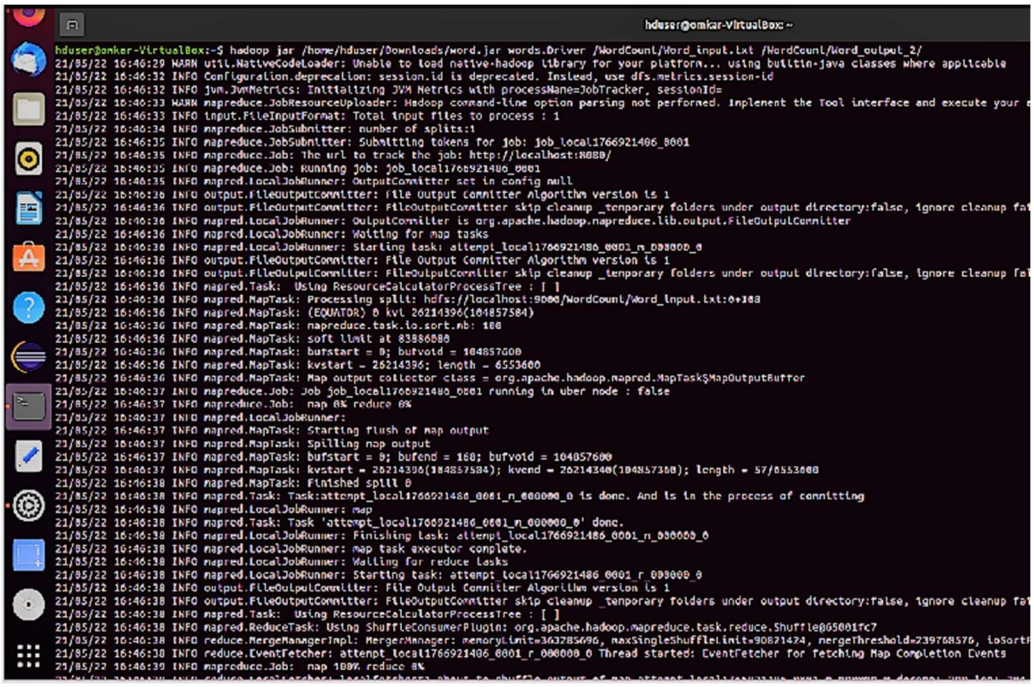
e.printStackTrace();

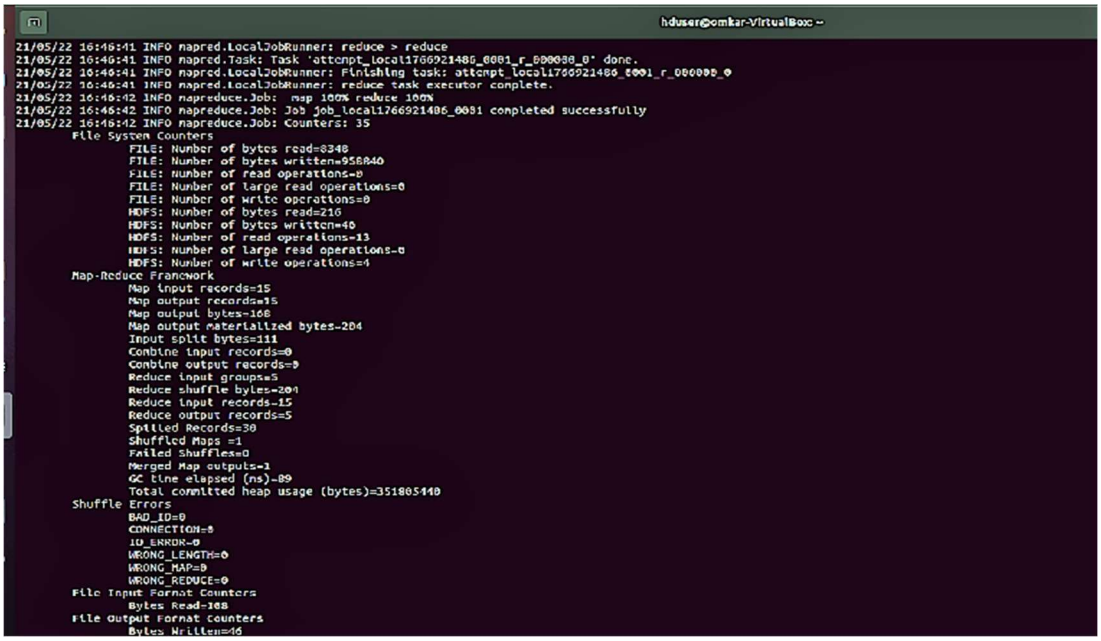
}

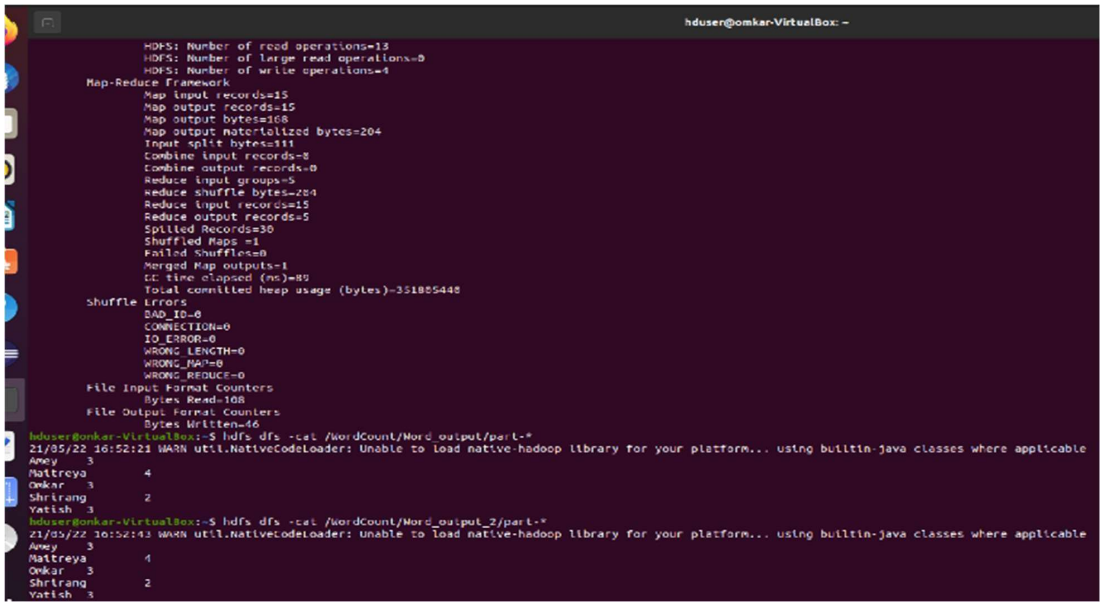
}

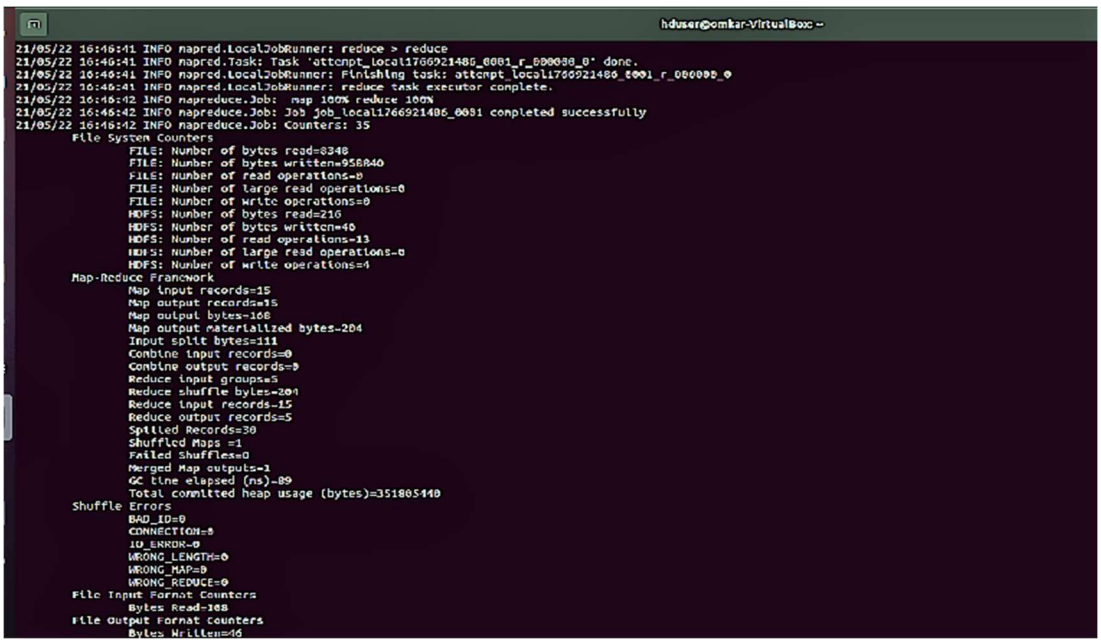
}

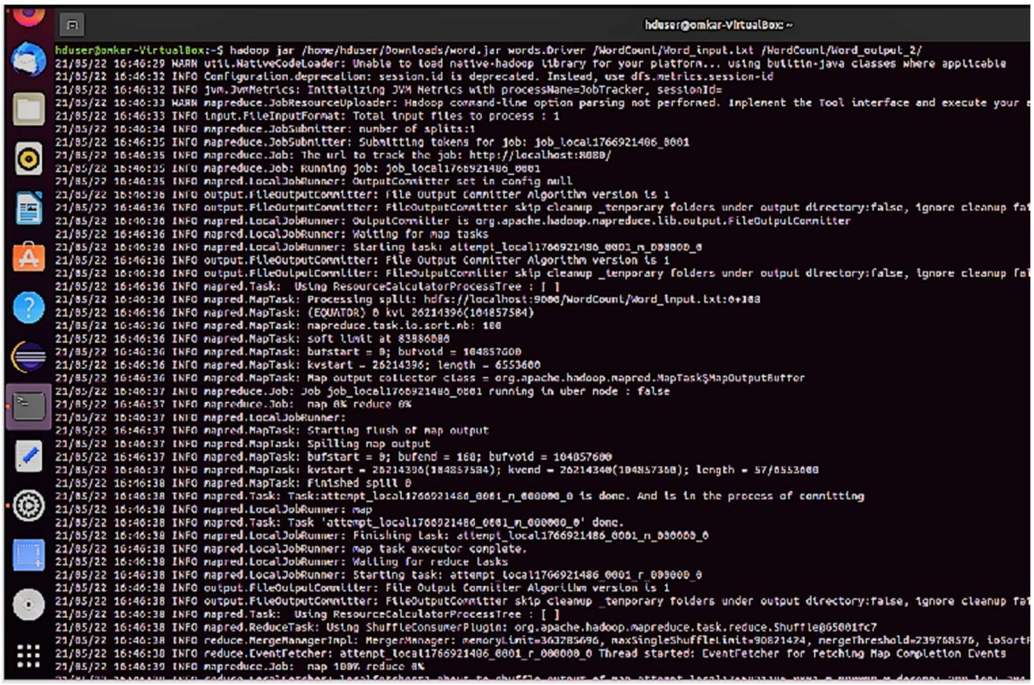
Output:

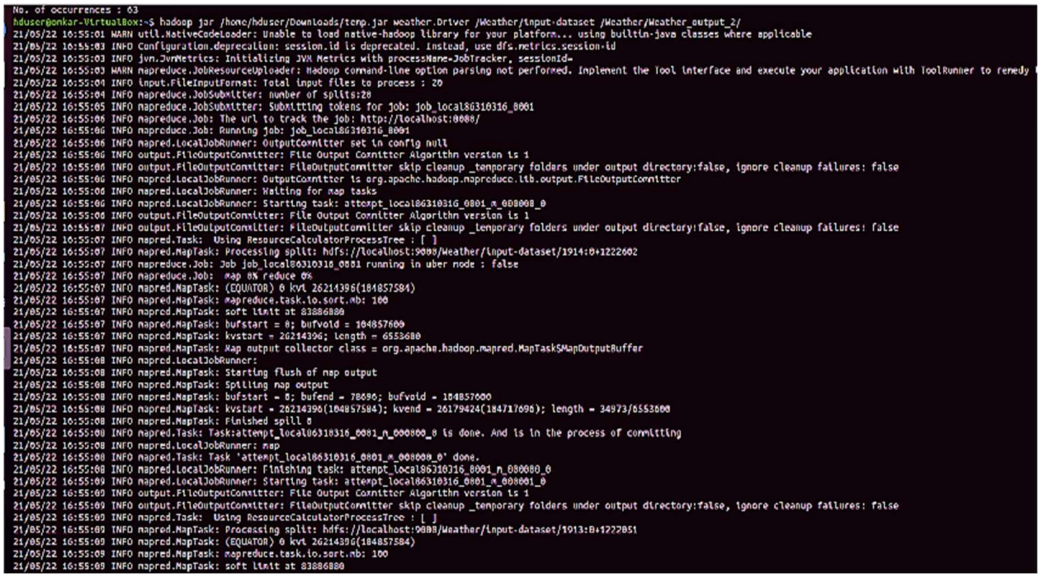



hduser@omkar-VirtualBox: ~
21/05/22 16:46:41 INFO mapred.LocalJobRunner: reduce > reduce
21/05/22 16:46:41 INFO mapred.Task: Task 'attempt_local1766921486_0001_r_000000_0' done.
21/05/22 16:46:41 INFO mapred.LocalJobRunner: Finishing task: attempt_local1766921486_0001_r_000000_0
21/05/22 16:46:41 INFO mapred.LocalJobRunner: reduce task executor complete.
21/05/22 16:46:42 INFO mapreduce.Job: map 100% reduce 100%
21/05/22 16:46:42 INFO mapreduce.Job: Job job_local1766921486_0001 completed successfully
21/05/22 16:46:42 INFO mapreduce.Job: Counters: 35
File System Counters
FILE: Number of bytes read=8348
FILE: Number of bytes written=958840
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=216
HDFS: Number of bytes written=40
HDFS: Number of read operations=13
HDFS: Number of large read operations=0
HDFS: Number of write operations=4
Map-Reduce Framework
Map input records=15
Map output records=15
Map output bytes=168
Map output materialized bytes=204
Input split bytes=111
Combine input records=0
Combine output records=0
Reduce input groups=5
Reduce shuffle bytes=204
Reduce input records=15
Reduce output records=5
Spilled Records=30
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=89
Total committed heap usage (bytes)=351805440
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=108
File Output Format Counters
Bytes Written=46


hduser@omkar-VirtualBox: ~
HDFS: Number of read operations=13
HDFS: Number of large read operations=0
HDFS: Number of write operations=4
Map-Reduce Framework
Map input records=15
Map output records=15
Map output bytes=168
Map output materialized bytes=204
Input split bytes=111
Combine input records=0
Combine output records=0
Reduce input groups=5
Reduce shuffle bytes=204
Reduce input records=15
Reduce output records=5
Spilled Records=30
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=89
Total committed heap usage (bytes)=351805440
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=108
File Output Format Counters
Bytes Written=46
hduser@omkar-VirtualBox:~$ hdfs dfs -cat /WordCount/Word_output/part-*
21/05/22 16:52:21 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Amey 3
Maitreya 4
Omkar 3
Shrirang 2
Yatish 3
hduser@omkar-VirtualBox:~$ hdfs dfs -cat /WordCount/Word_output_2/part-*
21/05/22 16:52:43 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Amey 3
Maitreya 4
Omkar 3
Shrirang 2
Yatish 3


hduser@omkar-VirtualBox: ~
21/05/22 16:46:41 INFO mapred.LocalJobRunner: reduce > reduce
21/05/22 16:46:41 INFO mapred.Task: Task 'attempt_local1766921486_0001_r_000000_0' done.
21/05/22 16:46:41 INFO mapred.LocalJobRunner: Finishing task: attempt_local1766921486_0001_r_000000_0
21/05/22 16:46:41 INFO mapred.LocalJobRunner: reduce task executor complete.
21/05/22 16:46:42 INFO mapreduce.Job: map 100% reduce 100%
21/05/22 16:46:42 INFO mapreduce.Job: Job job_local1766921486_0001 completed successfully
21/05/22 16:46:42 INFO mapreduce.Job: Counters: 35
File System Counters
FILE: Number of bytes read=8348
FILE: Number of bytes written=958840
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=216
HDFS: Number of bytes written=40
HDFS: Number of read operations=13
HDFS: Number of large read operations=0
HDFS: Number of write operations=4
Map-Reduce Framework
Map input records=15
Map output records=15
Map output bytes=168
Map output materialized bytes=204
Input split bytes=111
Combine input records=0
Combine output records=0
Reduce input groups=5
Reduce shuffle bytes=204
Reduce input records=15
Reduce output records=5
Spilled Records=30
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=89
Total committed heap usage (bytes)=351805440
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=108
File Output Format Counters
Bytes Written=46





Conclusion: Thus program using mareduce is implemented.

ASSIGNMENT NO: 4

Title:

Perform the following operations using Python on the Facebook metrics data sets

a. Create data subsets

b. Merge Data

c. Sort Data

d. Transposing Data

e. Shape and reshape Data

Problem Statement:

Study and perform different operations on dataset.

Theory:

What is python?

Python is a popular high-level programming language that was first released in 1991. It was created by Guido van Rossum and is named after the comedy group Monty Python. Python is known for its simplicity, readability, and versatility. It is an interpreted language, which means that code is executed line by line, rather than being compiled all at once.

Python has a large and supportive community, with a vast array of libraries and frameworks that make it useful for a variety of applications, from web development to data analysis and machine learning. Python is often considered an easy language for beginners to learn, but it is also a powerful language that is used by many professional developers and organizations.

Code & output:

Problem Statement

Perform the following operations using Python on the Facebook metrics data sets

1. Create data subsets
2. Merge Data
3. Sort Data
4. Transposing Data
5. Shape and reshape Data

Importing Libraries

In [1]:
```
import numpy as np
import pandas as pd
```

Importing the dataset

In [3]:
```
data = pd.read_csv('Datasets/dataset_Facebook.csv',sep=';')
data.head()
```

Out[3]:

	Page total likes	Type	Category	Post Month	Post Weekday	Post Hour	Paid	Lifetime Post Total Reach	Lifetime Post Total Impressions	Lifetime Engaged Users	L Con
0	139441	Photo	2	12	4	3	0.0	2752	5091	178	
1	139441	Status	2	12	3	10	0.0	10460	19057	1457	
2	139441	Photo	3	12	3	3	0.0	2413	4373	177	
3	139441	Photo	2	12	2	10	1.0	50128	87991	2211	
4	139441	Photo	2	12	2	3	0.0	7244	13594	671	

In [4]:
```
#Describing the data
data.describe()
```

Out[4]:

	Page total likes	Category	Post Month	Post Weekday	Post Hour	Paid	Lifetime Post Total Reach
count	500.000000	500.000000	500.000000	500.000000	500.000000	499.000000	500.00000
mean	123194.176000	1.880000	7.038000	4.150000	7.840000	0.278557	13903.36000
std	16272.813214	0.852675	3.307936	2.030701	4.368589	0.448739	22740.78789
min	81370.000000	1.000000	1.000000	1.000000	1.000000	0.000000	238.00000
25%	112676.000000	1.000000	4.000000	2.000000	3.000000	0.000000	3315.00000
50%	129600.000000	2.000000	7.000000	4.000000	9.000000	0.000000	5281.00000
75%	136393.000000	3.000000	10.000000	6.000000	11.000000	1.000000	13168.00000
max	139441.000000	3.000000	12.000000	7.000000	23.000000	1.000000	180480.00000

1) Creating Subsets

In [14]:
```
subset1 = data[['Page total likes','Category','Post Month','Post Weekday']].loc[0:
subset1
```

Out[14]:

	Page total likes	Category	Post Month	Post Weekday
0	139441	2	12	4
1	139441	2	12	3
2	139441	3	12	3
3	139441	2	12	2
4	139441	2	12	2
5	139441	2	12	1
6	139441	3	12	1
7	139441	3	12	7
8	139441	2	12	7
9	139441	3	12	6
10	139441	2	12	5
11	139441	2	12	5
12	139441	2	12	5
13	139441	2	12	5
14	138414	2	12	4
15	138414	2	12	3

In [15]:
```
subset2 = data[['Page total likes','Category','Post Month','Post Weekday']].loc[16
subset2
```

Out[15]:

	Page total likes	Category	Post Month	Post Weekday
16	138414	3	12	3
17	138414	1	12	2
18	138414	3	12	2
19	138414	3	12	1
20	138414	2	12	1
21	138414	1	12	7
22	138414	1	12	7
23	138414	3	12	7
24	138414	2	12	6
25	138458	2	12	6
26	138458	2	12	5
27	138458	3	12	5
28	138895	2	12	5
29	138895	1	12	4
30	138895	2	12	4

In [16]:
```
subset3 = data[['Page total likes','Category','Post Month','Post Weekday']].loc[31
subset3
```

Out[16]:

	Page total likes	Category	Post Month	Post Weekday
31	138895	2	12	3
32	138895	3	12	3
33	138895	3	12	2
34	138895	1	12	2
35	138895	2	12	1
36	138895	3	12	1
37	138895	1	12	7
38	138895	2	12	7
39	138895	1	12	7
40	138895	2	12	6
41	138895	1	12	6
42	138353	1	12	5
43	138353	1	12	5
44	138353	1	12	4
45	138353	1	12	4
46	138353	1	12	3
47	138353	1	12	3
48	138353	1	12	2
49	138353	1	12	2
50	138353	2	11	1

2) Merging data

In [17]:
```
mergedData=pd.concat([subset1,subset2,subset3])
mergedData
```

Out[17]:

	Page total likes	Category	Post Month	Post Weekday
0	139441	2	12	4
1	139441	2	12	3
2	139441	3	12	3
3	139441	2	12	2
4	139441	2	12	2
5	139441	2	12	1
6	139441	3	12	1
7	139441	3	12	7
8	139441	2	12	7
9	139441	3	12	6
10	139441	2	12	5
11	139441	2	12	5
12	139441	2	12	5
13	139441	2	12	5
14	138414	2	12	4
15	138414	2	12	3
16	138414	3	12	3
17	138414	1	12	2
18	138414	3	12	2
19	138414	3	12	1
20	138414	2	12	1
21	138414	1	12	7
22	138414	1	12	7
23	138414	3	12	7
24	138414	2	12	6
25	138458	2	12	6
26	138458	2	12	5
27	138458	3	12	5
28	138895	2	12	5
29	138895	1	12	4
30	138895	2	12	4
31	138895	2	12	3
32	138895	3	12	3
33	138895	3	12	2
34	138895	1	12	2
35	138895	2	12	1

	Page total likes	Category	Post Month	Post Weekday
36	138895	3	12	1
37	138895	1	12	7
38	138895	2	12	7
39	138895	1	12	7
40	138895	2	12	6
41	138895	1	12	6
42	138353	1	12	5
43	138353	1	12	5
44	138353	1	12	4
45	138353	1	12	4
46	138353	1	12	3
47	138353	1	12	3
48	138353	1	12	2
49	138353	1	12	2
50	138353	2	11	1

3) Sort data

In [19]:
```
sortedData=data.sort_values('Page total likes', ascending=False)
sortedData
```

Out[19]:

	Page total likes	Type	Category	Post Month	Post Weekday	Post Hour	Paid	Lifetime Post Total Reach	Lifetime Post Total Impressions	Lifetime Engaged Users	C
0	139441	Photo	2	12	4	3	0.0	2752	5091	178	
8	139441	Status	2	12	7	3	0.0	11844	22538	1530	
1	139441	Status	2	12	3	10	0.0	10460	19057	1457	
12	139441	Photo	2	12	5	10	0.0	2847	5133	193	
11	139441	Photo	2	12	5	10	0.0	3112	5590	208	
...	...	...	...	...	...	...	...	...	...	...	...
495	85093	Photo	3	1	7	2	0.0	4684	7536	733	
496	81370	Photo	2	1	5	8	0.0	3480	6229	537	
497	81370	Photo	1	1	5	2	0.0	3778	7216	625	
498	81370	Photo	3	1	4	11	0.0	4156	7564	626	
499	81370	Photo	2	1	4	4	NaN	4188	7292	564	

500 rows × 19 columns

4) Transposing data

In [20]:
```
transposedData=data.transpose()
transposedData
```

Out[20]:

	0	1	2	3	4	5	6	7	8	9
Page total likes	139441	139441	139441	139441	139441	139441	139441	139441	139441	139441
Type	Photo	Status	Photo	Photo	Photo	Status	Photo	Photo	Status	Photo
Category	2	2	3	2	2	2	3	3	2	3
Post Month	12	12	12	12	12	12	12	12	12	12
Post Weekday	4	3	3	2	2	1	1	7	7	6
Post Hour	3	10	3	10	3	9	3	9	3	10
Paid	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0
Lifetime Post Total Reach	2752	10460	2413	50128	7244	10472	11692	13720	11844	4694
Lifetime Post Total Impressions	5091	19057	4373	87991	13594	20849	19479	24137	22538	8668
Lifetime Engaged Users	178	1457	177	2211	671	1191	481	537	1530	280
Lifetime Post Consumers	109	1361	113	790	410	1073	265	232	1407	183
Lifetime Post Consumptions	159	1674	154	1119	580	1389	364	305	1692	250
Lifetime Post Impressions by people who have liked your Page	3078	11710	2812	61027	6228	16034	15432	19728	15220	4309
Lifetime Post reach by people who like your Page	1640	6112	1503	32048	3200	7852	9328	11056	7912	2324
Lifetime People who have liked your Page and engaged with your post	119	1108	132	1386	396	1016	379	422	1250	199
comment	4	5	0	58	19	1	3	0	0	3
like	79.0	130.0	66.0	1572.0	325.0	152.0	249.0	325.0	161.0	113.0
share	17.0	29.0	14.0	147.0	49.0	33.0	27.0	14.0	31.0	26.0
Total Interactions	100	164	80	1777	393	186	279	339	192	142

19 rows × 500 columns

5) Shape and reshape data

In [6]:
```
nrows=data.shape[0]
ncols=data.shape[1]
print(f'The dataset consists of {nrows} rows and {ncols} columns.')
```

```
The dataset consists of 500 rows and 19 columns.
```

In [22]:
```
# Reshaping
pivotTable = pd.pivot_table(data,index=['Type','Category'],values='comment')
pivotTable
```

Out[22]:

		comment
Type	**Category**	
Link	**1**	2.900000
	2	2.000000
	3	2.000000
Photo	**1**	5.897297
	2	11.692308
	3	6.913333
Status	**1**	4.333333
	2	9.921053
	3	2.750000
Video	**1**	12.285714

In [24]:
```
# Reshaping an array
array = np.arange(1,11)
reshapedArray = array.reshape(5,2)
reshapedArray
```

Out[24]:
```
array([[ 1,  2],
       [ 3,  4],
       [ 5,  6],
       [ 7,  8],
       [ 9, 10]])
```

ASSIGNMENT NO: 5

Title:

Perform the following operations using Python on the Air quality and Heart Diseases data sets
a. Data cleaning

b. Data integration

c. Data transformation

d. Error correcting

e. Data model building

Problem Statement:

Study and perform different operations on dataset.

Theory:

Data Cleaning is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data.

Data Transformation:

Stages of Predictive Modeling

Predictive Modeling is the process of building a model to predict future outcomes using statistics techniques. In order to generate the model, historical data of prior occurrences needs to be analyzed, classified and validated.

Listed below are the stages of predictive modeling:

1. Data Gathering and Cleansing
2. Data Analysis/Transformation
3. Building a Predictive Model
4. Inferences

Air Pollution In India

Air pollution in India is a serious issue with the major sources being fuelwood and biomass burning, fuel adulteration, vehicle emission and traffic congestion. In autumn and winter months, large scale crop residue burning in agriculture fields – a low cost alternative to mechanical tilling – is a major source of smoke, smog and particulate pollution. India has a low per capita emissions of greenhouse gases but the country as a whole is the third largest after China and the United States.A 2013 study on non-smokers has found that Indians have 30% lower lung function compared to Europeans.

The Air (Prevention and Control of Pollution) Act was passed in 1981 to regulate air pollution and there have been some measurable improvements. However, the 2016 Environmental Performance Index ranked India 141 out of 180 countries.

Pollution is turning the Taj Mahal yellow, despite efforts by the Indian government to control air contamination around the poignant 17th century monument and keep it shimmering white, a parliamentary committee has said.

In a report to parliament this week, the standing committee on transport, tourism and culture said airborne particles were being deposited on the monument's white marble, giving it a yellow tinge.

The monument, in the northern city of Agra about four hours drive south of the capital, was built by Mughal emperor Shah Jahan as a mausoleum for his wife Mumtaz Mahal.

Authorities have made various attempts in the past to keep the area around the Taj Mahal pollution free, including setting up an air pollution monitoring station in Agra.

But the committee said that while air pollutants such as sulphur dioxide and nitrous oxide gases were generally within permissible limits, "suspended particulate matter" had been recorded at high

Code & output:

In [1]:
```
#Importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
#Loading csv file on a dataframe

ar = pd.read_csv('Datasets/AirQuality.csv',sep=';')
ar.head()
```

Out[3]:

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)
0	10/03/2004	18.00.00	2,6	1360.0	150.0	11,9	1046.0	166.0
1	10/03/2004	19.00.00	2	1292.0	112.0	9,4	955.0	103.0
2	10/03/2004	20.00.00	2,2	1402.0	88.0	9,0	939.0	131.0
3	10/03/2004	21.00.00	2,2	1376.0	80.0	9,2	948.0	172.0
4	10/03/2004	22.00.00	1,6	1272.0	51.0	6,5	836.0	131.0

In [4]:
```
ar.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9471 entries, 0 to 9470
Data columns (total 17 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Date           9357 non-null   object
 1   Time           9357 non-null   object
 2   CO(GT)         9357 non-null   object
 3   PT08.S1(CO)    9357 non-null   float64
 4   NMHC(GT)       9357 non-null   float64
 5   C6H6(GT)       9357 non-null   object
 6   PT08.S2(NMHC)  9357 non-null   float64
 7   NOx(GT)        9357 non-null   float64
 8   PT08.S3(NOx)   9357 non-null   float64
 9   NO2(GT)        9357 non-null   float64
 10  PT08.S4(NO2)   9357 non-null   float64
 11  PT08.S5(O3)    9357 non-null   float64
 12  T              9357 non-null   object
 13  RH             9357 non-null   object
 14  AH             9357 non-null   object
 15  Unnamed: 15    0 non-null      float64
 16  Unnamed: 16    0 non-null      float64
dtypes: float64(10), object(7)
memory usage: 1.2+ MB
```

In [5]:
```
#Dropping CO(GT) and Unnamed columns
ar.drop(['CO(GT)','Unnamed: 15','Unnamed: 16'],axis = 1,inplace = True)
```

In [6]:
```
#Formatting some object columns from strings to floats

ar.replace(to_replace=',',value='.',regex=True,inplace=True)

for i in 'C6H6(GT) T RH AH'.split():
    ar[i] = pd.to_numeric(ar[i],errors='coerce')
```

In [7]:
```
#Replacing null data from -200 to NaN for posterior treatment

ar.replace(to_replace=-200,value=np.nan,inplace=True)

ar.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9471 entries, 0 to 9470
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Date           9357 non-null   object 
 1   Time           9357 non-null   object 
 2   PT08.S1(CO)    8991 non-null   float64
 3   NMHC(GT)       914 non-null    float64
 4   C6H6(GT)       8991 non-null   float64
 5   PT08.S2(NMHC)  8991 non-null   float64
 6   NOx(GT)        7718 non-null   float64
 7   PT08.S3(NOx)   8991 non-null   float64
 8   NO2(GT)        7715 non-null   float64
 9   PT08.S4(NO2)   8991 non-null   float64
 10  PT08.S5(O3)    8991 non-null   float64
 11  T              8991 non-null   float64
 12  RH             8991 non-null   float64
 13  AH             8991 non-null   float64
dtypes: float64(12), object(2)
memory usage: 1.0+ MB
```

In [8]:
```
#Formatting Date and Time to datetime type

ar['Date'] = pd.to_datetime(ar['Date'],dayfirst=True)

ar['Time'] = pd.to_datetime(ar['Time'],format= '%H.%M.%S' ).dt.time

ar.head()
```

Out[8]:

	Date	Time	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)	PT08.S3(NOx)
0	2004-03-10	18:00:00	1360.0	150.0	11.9	1046.0	166.0	1056.0
1	2004-03-10	19:00:00	1292.0	112.0	9.4	955.0	103.0	1174.0
2	2004-03-10	20:00:00	1402.0	88.0	9.0	939.0	131.0	1140.0
3	2004-03-10	21:00:00	1376.0	80.0	9.2	948.0	172.0	1092.0
4	2004-03-10	22:00:00	1272.0	51.0	6.5	836.0	131.0	1205.0

In [9]:
```
NMHC_ratio = ar['NMHC(GT)'].isna().sum()/len(ar['NMHC(GT)'])

print('The NMHC(GT) sensor has {:.2f}% of missing data.'.format(NMHC_ratio*100))
```

```
The NMHC(GT) sensor has 90.35% of missing data.
```

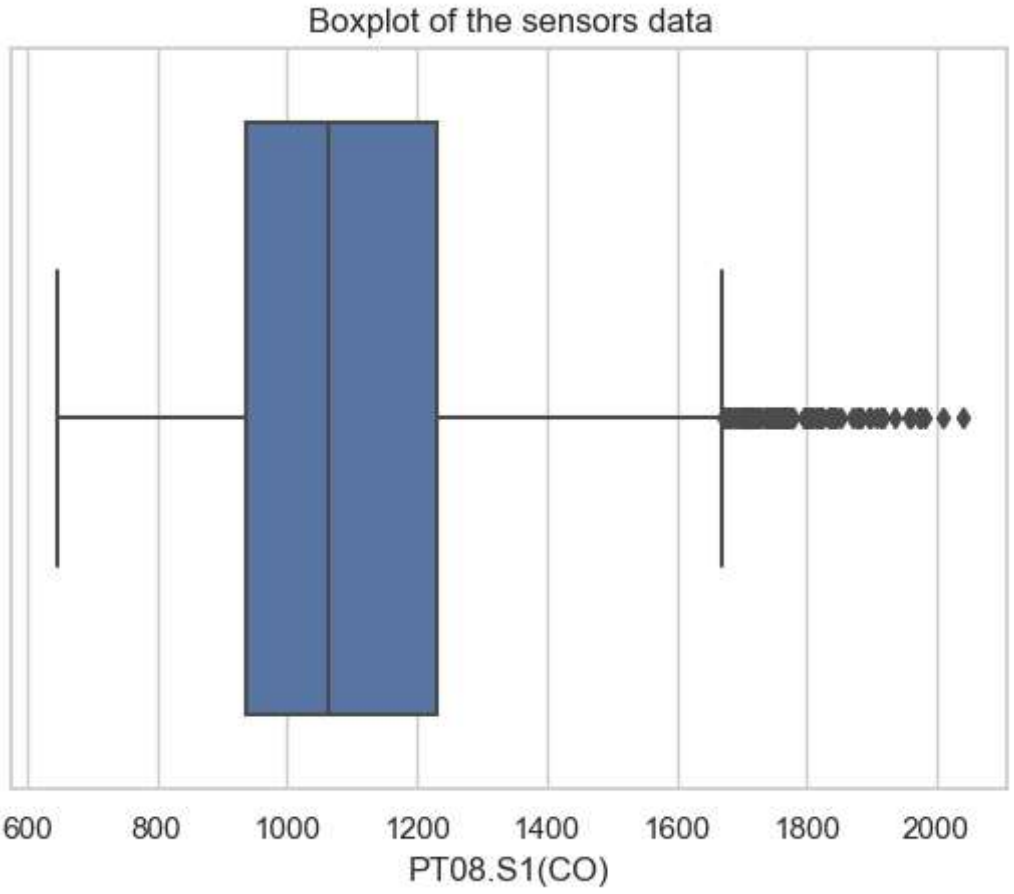
In [10]:
```
#Removing NMHC(GT) sensor due to amount of null values

ar.drop('NMHC(GT)', axis=1, inplace=True)
```

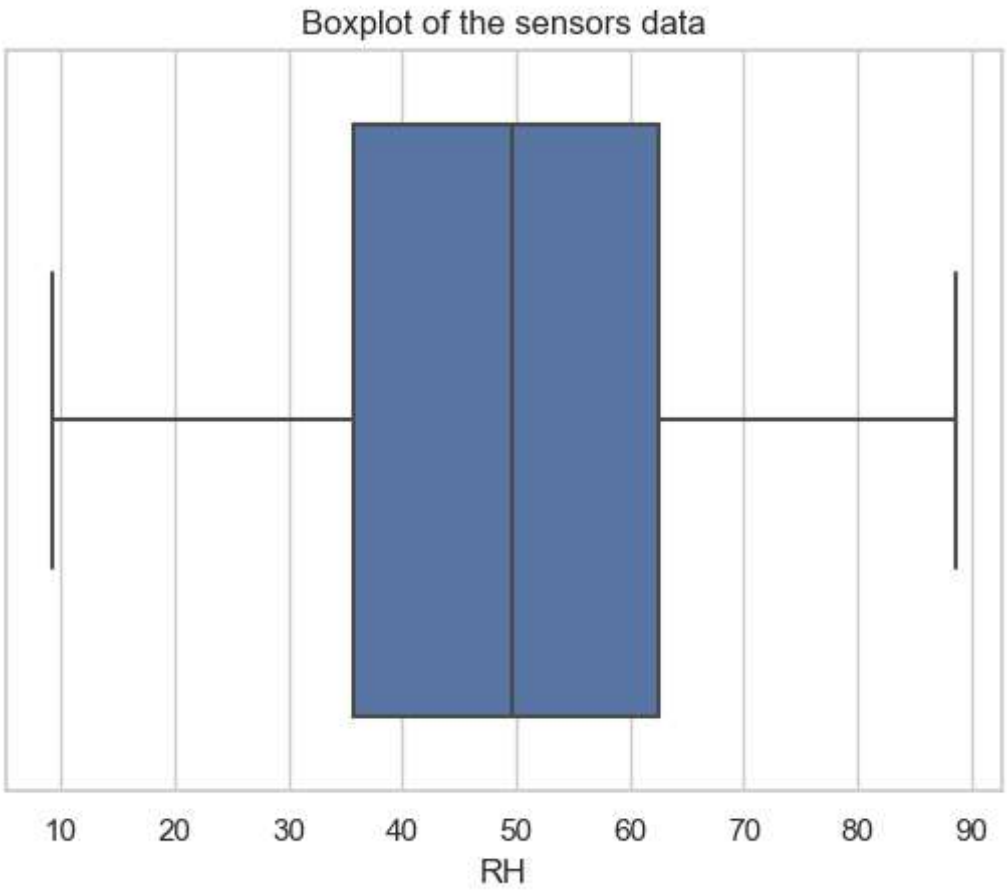
```
ar.info()
```

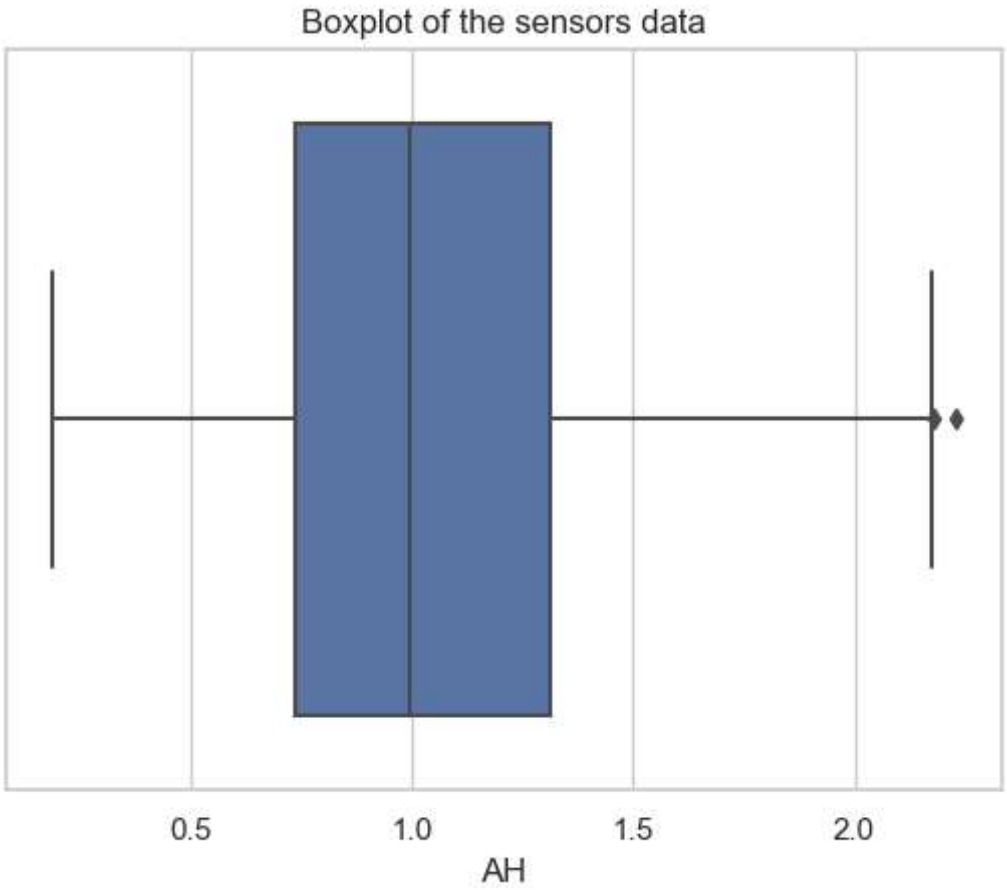
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9471 entries, 0 to 9470
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Date           9357 non-null   datetime64[ns]
 1   Time           9357 non-null   object
 2   PT08.S1(CO)    8991 non-null   float64
 3   C6H6(GT)       8991 non-null   float64
 4   PT08.S2(NMHC)  8991 non-null   float64
 5   NOx(GT)        7718 non-null   float64
 6   PT08.S3(NOx)   8991 non-null   float64
 7   NO2(GT)        7715 non-null   float64
 8   PT08.S4(NO2)   8991 non-null   float64
 9   PT08.S5(O3)    8991 non-null   float64
 10  T              8991 non-null   float64
 11  RH             8991 non-null   float64
 12  AH             8991 non-null   float64
dtypes: datetime64[ns](1), float64(11), object(1)
memory usage: 962.0+ KB
```

In [11]:
```
sns.set_theme(style="whitegrid")
```

In [12]:
```
for i in ar.columns[2:13]:
    sns.boxplot(x=ar[i])
    plt.title('Boxplot of the sensors data')
    plt.show()
```







In [13]:
```
#Removing Outliers with the Interquartile Range Method (IQR)

Q1 = ar.quantile(0.25) #first 25% of the data
Q3 = ar.quantile(0.75) #first 75% of the data
IQR = Q3 - Q1 #IQR = InterQuartile Range
```

```
scale = 2 #For Normal Distributions, scale = 1.5
lower_lim = Q1 - scale*IQR
upper_lim = Q3 + scale*IQR

lower_outliers = (ar[ar.columns[2:13]] < lower_lim)
upper_outliers = (ar[ar.columns[2:13]] > upper_lim)
```

```
C:\Users\ASUS\AppData\Local\Temp\ipykernel_17060\3919401421.py:3: FutureWarning: T
he default value of numeric_only in DataFrame.quantile is deprecated. In a future
version, it will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  Q1 = ar.quantile(0.25) #first 25% of the data
C:\Users\ASUS\AppData\Local\Temp\ipykernel_17060\3919401421.py:4: FutureWarning: T
he default value of numeric_only in DataFrame.quantile is deprecated. In a future
version, it will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  Q3 = ar.quantile(0.75) #first 75% of the data
```

In [14]:
```
#Checking the resulting outliers calculated by the above method (represented below

ar[ar.columns[2:13]][(lower_outliers | upper_outliers)].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9471 entries, 0 to 9470
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   PT08.S1(CO)    30 non-null     float64
 1   C6H6(GT)       105 non-null    float64
 2   PT08.S2(NMHC)  13 non-null     float64
 3   NOx(GT)        244 non-null    float64
 4   PT08.S3(NOx)   114 non-null    float64
 5   NO2(GT)        32 non-null     float64
 6   PT08.S4(NO2)   20 non-null     float64
 7   PT08.S5(O3)    17 non-null     float64
 8   T              0 non-null      float64
 9   RH             0 non-null      float64
 10  AH             0 non-null      float64
dtypes: float64(11)
memory usage: 814.0 KB
```

In [15]:
```
#Create new DataFrame without the outliers

num_cols = list(ar.columns[2:13])
ar_out_IQR = ar[~((ar[num_cols] < (Q1 - 2 * IQR)) |(ar[num_cols] > (Q3 + 2 * IQR))
ar_out_IQR.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9029 entries, 0 to 9470
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Date           8915 non-null   datetime64[ns]
 1   Time           8915 non-null   object
 2   PT08.S1(CO)    8595 non-null   float64
 3   C6H6(GT)       8595 non-null   float64
 4   PT08.S2(NMHC)  8595 non-null   float64
 5   NOx(GT)        7313 non-null   float64
 6   PT08.S3(NOx)   8595 non-null   float64
 7   NO2(GT)        7310 non-null   float64
 8   PT08.S4(NO2)   8595 non-null   float64
 9   PT08.S5(O3)    8595 non-null   float64
 10  T              8595 non-null   float64
 11  RH             8595 non-null   float64
 12  AH             8595 non-null   float64
dtypes: datetime64[ns](1), float64(11), object(1)
memory usage: 987.5+ KB
```

In [16]:
```
#Removing NOx(GT) and NO2(GT) sensor data due the ammount of null values if compare

pd.options.mode.chained_assignment = None
ar_out_IQR.drop(['NOx(GT)','NO2(GT)'],axis=1, inplace=True)
ar_out_IQR.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9029 entries, 0 to 9470
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Date           8915 non-null   datetime64[ns]
 1   Time           8915 non-null   object
 2   PT08.S1(CO)    8595 non-null   float64
 3   C6H6(GT)       8595 non-null   float64
 4   PT08.S2(NMHC)  8595 non-null   float64
 5   PT08.S3(NOx)   8595 non-null   float64
 6   PT08.S4(NO2)   8595 non-null   float64
 7   PT08.S5(O3)    8595 non-null   float64
 8   T              8595 non-null   float64
 9   RH             8595 non-null   float64
 10  AH             8595 non-null   float64
dtypes: datetime64[ns](1), float64(9), object(1)
memory usage: 846.5+ KB
```

In [17]:
```
#Eliminating rows with NaN values

ar_filt = ar_out_IQR.dropna(how='any', axis=0)
ar_filt.reset_index(drop=True,inplace=True)
```

In [18]:
```
ar_filt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8595 entries, 0 to 8594
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype         
---  ------          --------------  -----         
 0   Date            8595 non-null   datetime64[ns]
 1   Time            8595 non-null   object        
 2   PT08.S1(CO)     8595 non-null   float64       
 3   C6H6(GT)        8595 non-null   float64       
 4   PT08.S2(NMHC)   8595 non-null   float64       
 5   PT08.S3(NOx)    8595 non-null   float64       
 6   PT08.S4(NO2)    8595 non-null   float64       
 7   PT08.S5(O3)     8595 non-null   float64       
 8   T               8595 non-null   float64       
 9   RH              8595 non-null   float64       
 10  AH              8595 non-null   float64       
dtypes: datetime64[ns](1), float64(9), object(1)
memory usage: 738.8+ KB
```

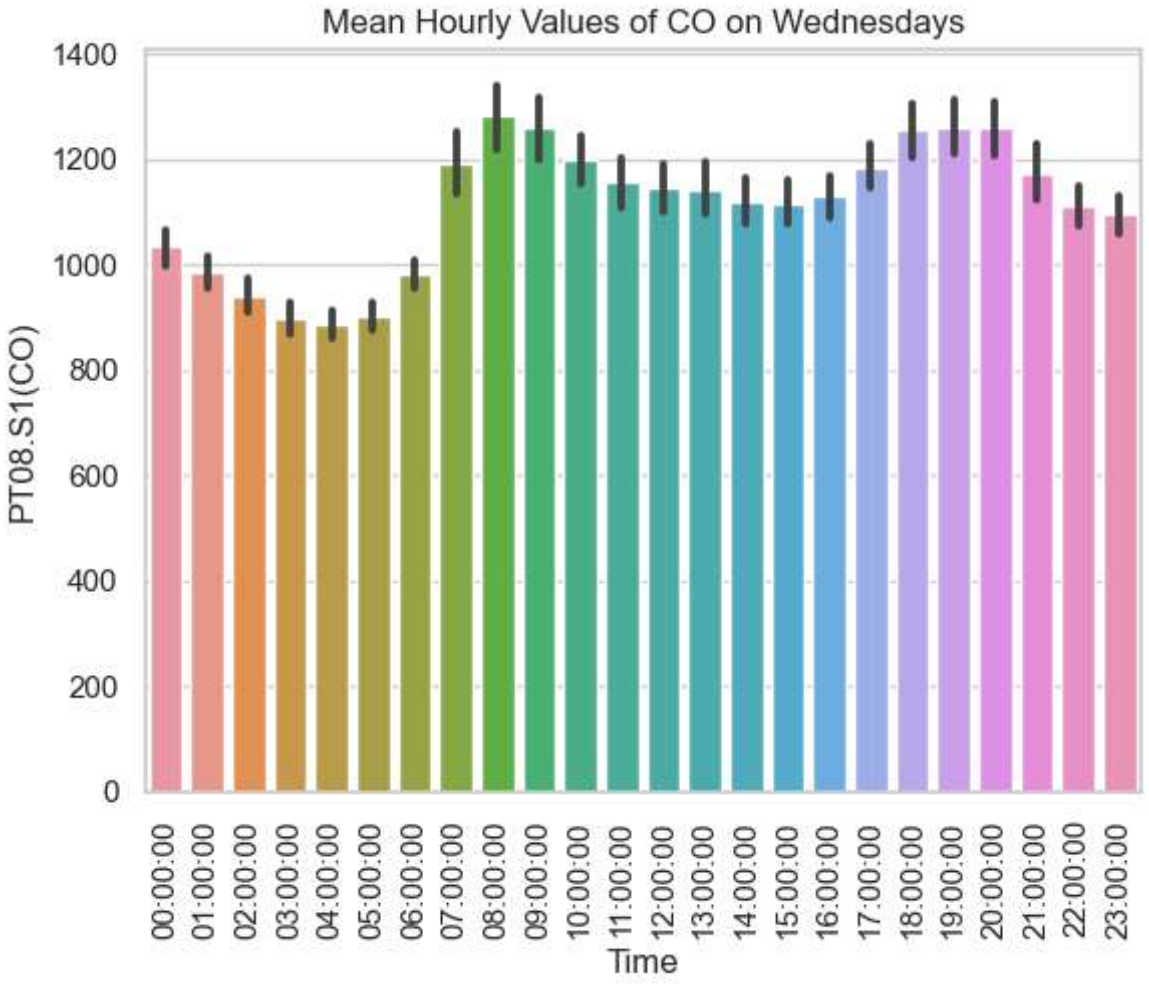
In [19]:
```
#Adding a column with the week days

ar_filt['Week Day'] = ar_filt['Date'].dt.day_name()

#Rearranging columns

cols = ar_filt.columns.tolist()
cols = cols[:1] + cols[-1:] + cols[1:11]
ar_filt = ar_filt[cols]
ar_filt.head(10)
```

Out[19]:

	Date	Week Day	Time	PT08.S1(CO)	C6H6(GT)	PT08.S2(NMHC)	PT08.S3(NOx)	PT08.S4(N
0	2004-03-10	Wednesday	18:00:00	1360.0	11.9	1046.0	1056.0	16
1	2004-03-10	Wednesday	19:00:00	1292.0	9.4	955.0	1174.0	15
2	2004-03-10	Wednesday	20:00:00	1402.0	9.0	939.0	1140.0	15
3	2004-03-10	Wednesday	21:00:00	1376.0	9.2	948.0	1092.0	15
4	2004-03-10	Wednesday	22:00:00	1272.0	6.5	836.0	1205.0	14
5	2004-03-10	Wednesday	23:00:00	1197.0	4.7	750.0	1337.0	13
6	2004-03-11	Thursday	00:00:00	1185.0	3.6	690.0	1462.0	13
7	2004-03-11	Thursday	01:00:00	1136.0	3.3	672.0	1453.0	13
8	2004-03-11	Thursday	02:00:00	1094.0	2.3	609.0	1579.0	12
9	2004-03-11	Thursday	07:00:00	1144.0	3.2	667.0	1490.0	13

In [20]:
```
#Creating new dataframe with only wednesday data
```

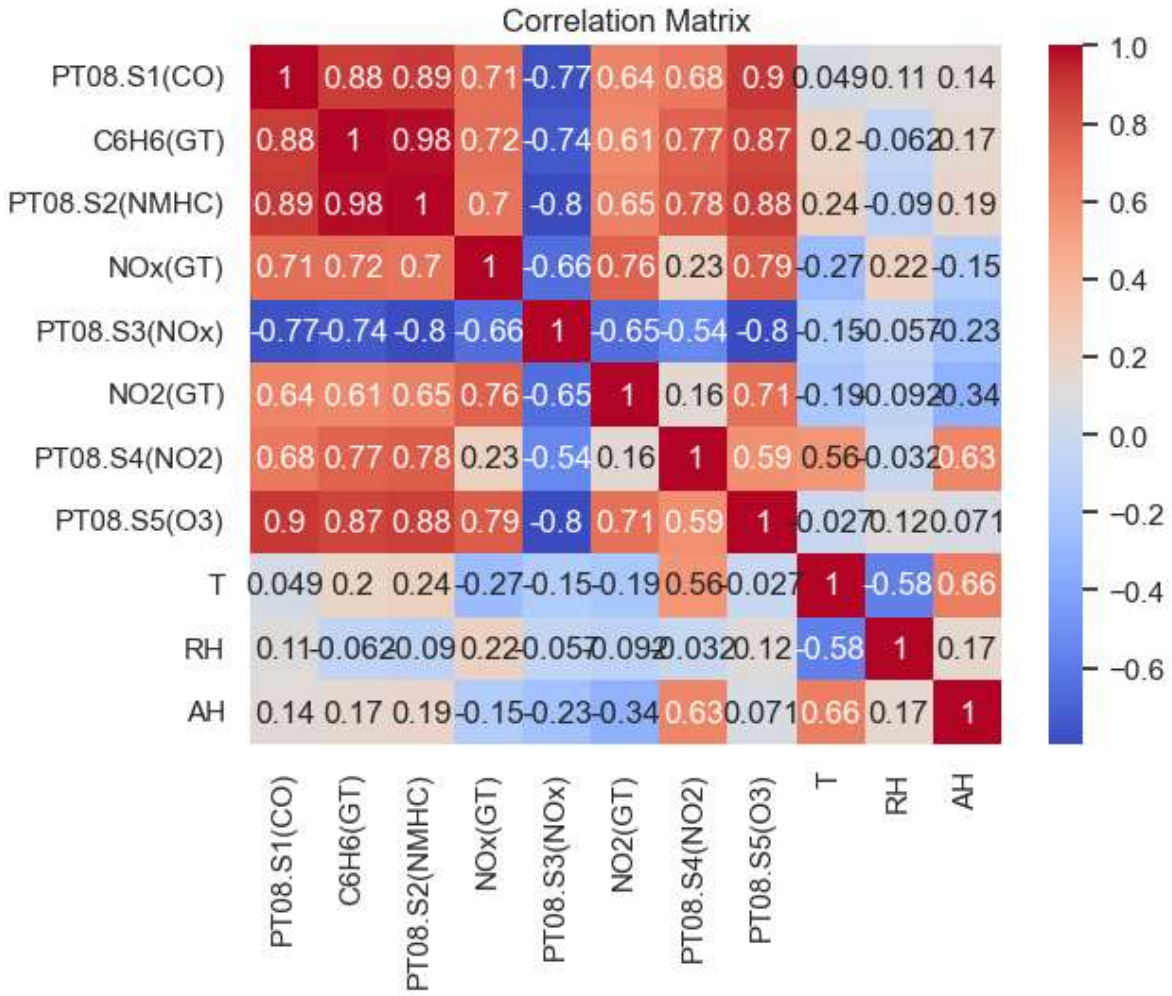
```
ar_wed = ar_filt[ar_filt['Week Day'] == 'Wednesday']

#Plotting the mean hourly value of CO on Wednesdays

sns.barplot(x='Time',y='PT08.S1(CO)', data=ar_wed.sort_values('Time'))
plt.title('Mean Hourly Values of CO on Wednesdays')
plt.xticks(rotation=90)
plt.show()
```



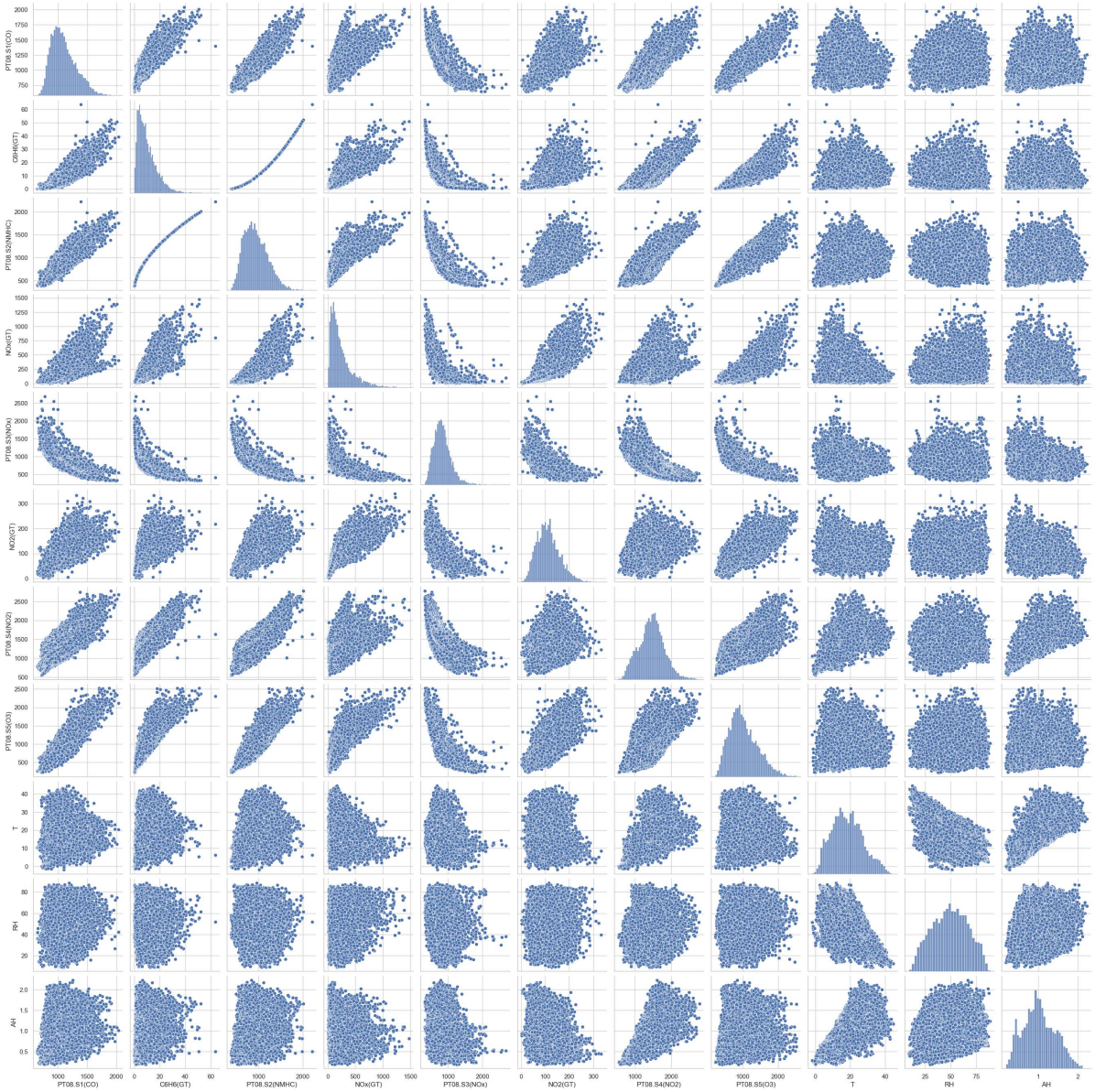
In [21]:
```
#The peak concentration of CO in the city are between 8 AM and 9 AM and between 6 P
#beginnings and endings of office hours, respectively.
```

In [22]:
```
#Plotting correlation matrix

sns.heatmap(ar.corr(),annot=True,cmap = 'coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

```
C:\Users\ASUS\AppData\Local\Temp\ipykernel_17060\2975330071.py:3: FutureWarning: T
he default value of numeric_only in DataFrame.corr is deprecated. In a future vers
ion, it will default to False. Select only valid columns or specify the value of n
umeric_only to silence this warning.
  sns.heatmap(ar.corr(),annot=True,cmap = 'coolwarm')
```



In [23]:
```
sns.pairplot(ar)
plt.show()
```



In [24]:
```
#Eliminating the C6H6(GT) for being a redundant sensor (the C6HC molecule is a Non-
#the correlation between those two sensor is exact

ar_filt.drop('C6H6(GT)', axis=1, inplace=True)
```

In [25]:
```
#Creating a Regression Model of the PT08.S1 sensor:
```

In [26]:
```
#Splitting the dataset in 80% for training and 20% for testing

from sklearn.model_selection import train_test_split

Y = ar_filt['PT08.S1(CO)'] #variável de predição
X = ar_filt.drop(['PT08.S1(CO)','Date', 'Time', 'Week Day'], axis=1)

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_st
print(X_train.shape, X_test.shape)
```

```
(6876, 7) (1719, 7)
```

In [27]:
```
from sklearn.ensemble import RandomForestRegressor

modelo_randomforest = RandomForestRegressor(n_estimators=100)

modelo_randomforest.fit(X_train, Y_train)
```

Out[27]:
```
▾ RandomForestRegressor
RandomForestRegressor()
```

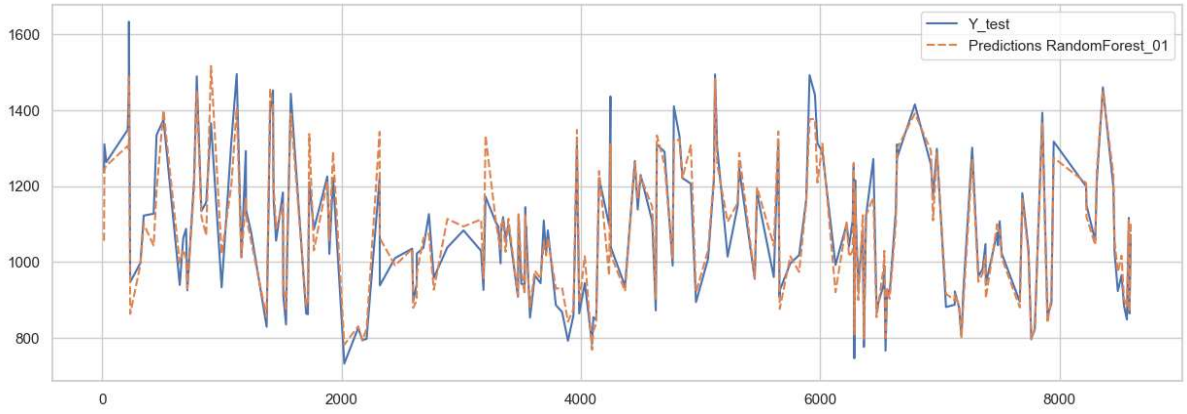
In [28]:
```
#Evaluating the results with the R² metric:
```

In [29]:
```
#Test data evaluation

from sklearn import metrics

pred_randomforest = modelo_randomforest.predict(X_test) #predicted CO concentratio

print('Random Forest Regression Model: R²={:.2f}'.format(metrics.r2_score(Y_test,
```

```
Random Forest Regression Model: R²=0.92
```

In [30]:
```
#Comparing model predictions with the Ground Truth Test Data
```

In [31]:
```
aux = pd.DataFrame()

aux['Y_test'] = Y_test
aux['Predictions RandomForest_01'] = pred_randomforest

plt.figure(figsize=(15,5))
sns.lineplot(data=aux.iloc[:200,:])
plt.show()
```



In [32]:
```
#Extracting season information from Date column:

def season(date):
    year = str(date.year)
    seasons = {'spring': pd.date_range(start='21/03/'+year, end='20/06/'+year),
               'summer': pd.date_range(start='21/06/'+year, end='22/09/'+year),
               'autumn': pd.date_range(start='23/09/'+year, end='20/12/'+year)}
    if date in seasons['spring']:
        return 'spring'
    if date in seasons['summer']:
        return 'summer'
    if date in seasons['autumn']:
        return 'autumn'
    else:
        return 'winter'

ar_filt['Season'] = ar_filt['Date'].map(season)
ar_filt.head(10)
```

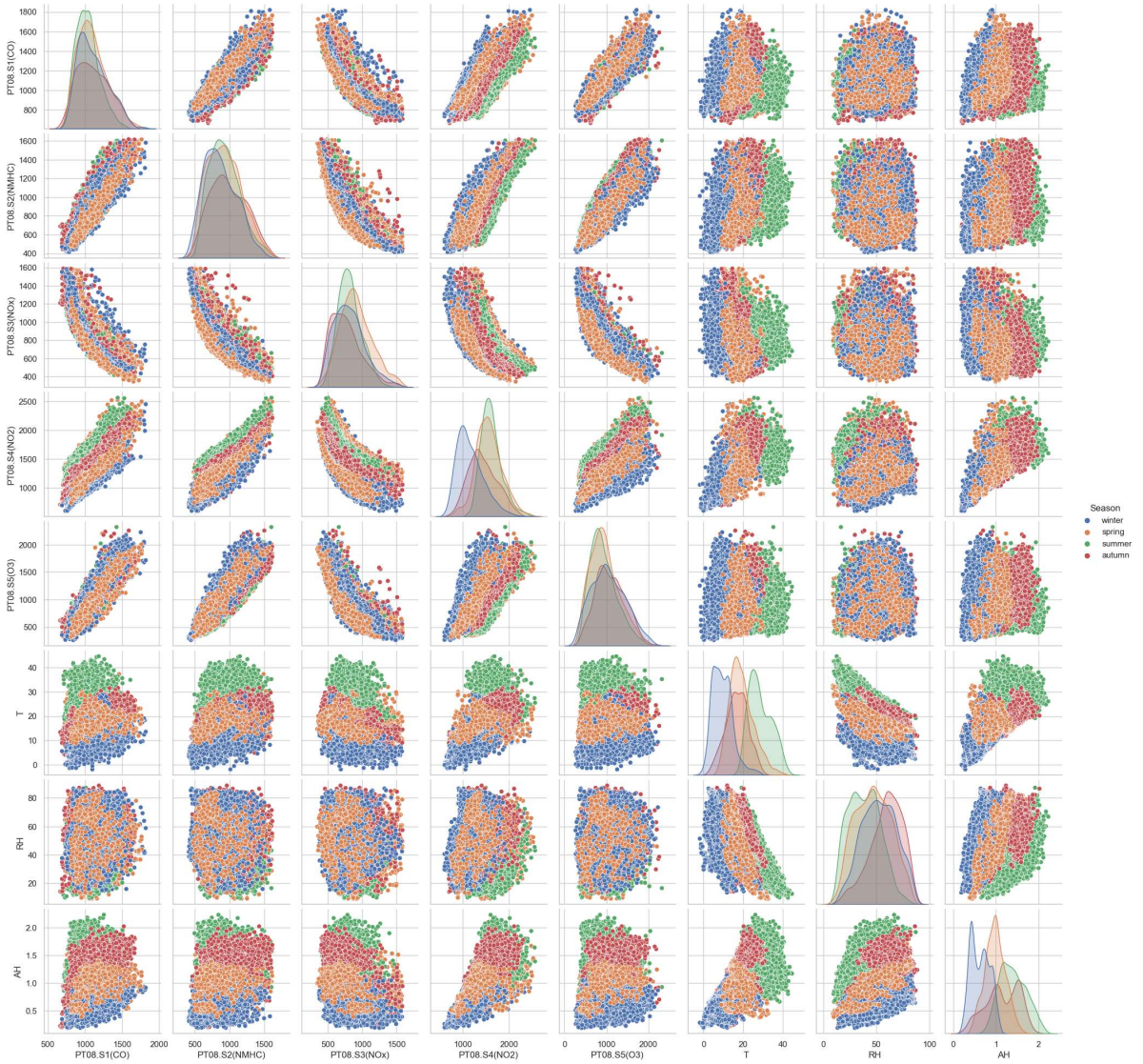
```
C:\Users\ASUS\AppData\Local\Temp\ipykernel_17060\1681381420.py:5: UserWarning: Par
sing dates in DD/MM/YYYY format when dayfirst=False (the default) was specified. T
his may lead to inconsistently parsed dates! Specify a format to ensure consistent
parsing.
  seasons = {'spring': pd.date_range(start='21/03/'+year, end='20/06/'+year),
C:\Users\ASUS\AppData\Local\Temp\ipykernel_17060\1681381420.py:6: UserWarning: Par
sing dates in DD/MM/YYYY format when dayfirst=False (the default) was specified. T
his may lead to inconsistently parsed dates! Specify a format to ensure consistent
parsing.
  'summer': pd.date_range(start='21/06/'+year, end='22/09/'+year),
C:\Users\ASUS\AppData\Local\Temp\ipykernel_17060\1681381420.py:7: UserWarning: Par
sing dates in DD/MM/YYYY format when dayfirst=False (the default) was specified. T
his may lead to inconsistently parsed dates! Specify a format to ensure consistent
parsing.
  'autumn': pd.date_range(start='23/09/'+year, end='20/12/'+year)}
```

Out[32]:

	Date	Week Day	Time	PT08.S1(CO)	PT08.S2(NMHC)	PT08.S3(NOx)	PT08.S4(NO2)	PT08.
0	2004-03-10	Wednesday	18:00:00	1360.0	1046.0	1056.0	1692.0	
1	2004-03-10	Wednesday	19:00:00	1292.0	955.0	1174.0	1559.0	
2	2004-03-10	Wednesday	20:00:00	1402.0	939.0	1140.0	1555.0	
3	2004-03-10	Wednesday	21:00:00	1376.0	948.0	1092.0	1584.0	
4	2004-03-10	Wednesday	22:00:00	1272.0	836.0	1205.0	1490.0	
5	2004-03-10	Wednesday	23:00:00	1197.0	750.0	1337.0	1393.0	
6	2004-03-11	Thursday	00:00:00	1185.0	690.0	1462.0	1333.0	
7	2004-03-11	Thursday	01:00:00	1136.0	672.0	1453.0	1333.0	
8	2004-03-11	Thursday	02:00:00	1094.0	609.0	1579.0	1276.0	
9	2004-03-11	Thursday	07:00:00	1144.0	667.0	1490.0	1339.0	

In [33]:

```
sns.pairplot(ar_filt, hue='Season')
plt.show()
```



In [34]:
```
#Creating categorical features from Season column and splitting new dataframe

Y_2 = ar_filt['PT08.S1(CO)']
X_2 = ar_filt.drop(['PT08.S1(CO)','Date', 'Time', 'Week Day'], axis=1)
X_2 = pd.get_dummies(data=X_2)

X_2_train, X_2_test, Y_2_train, Y_2_test = train_test_split(X_2, Y_2, test_size=0.
X_2.head()
```

Out[34]:

	PT08.S2(NMHC)	PT08.S3(NOx)	PT08.S4(NO2)	PT08.S5(O3)	T	RH	AH	Season_autumn
0	1046.0	1056.0	1692.0	1268.0	13.6	48.9	0.7578	0
1	955.0	1174.0	1559.0	972.0	13.3	47.7	0.7255	0
2	939.0	1140.0	1555.0	1074.0	11.9	54.0	0.7502	0
3	948.0	1092.0	1584.0	1203.0	11.0	60.0	0.7867	0
4	836.0	1205.0	1490.0	1110.0	11.2	59.6	0.7888	0

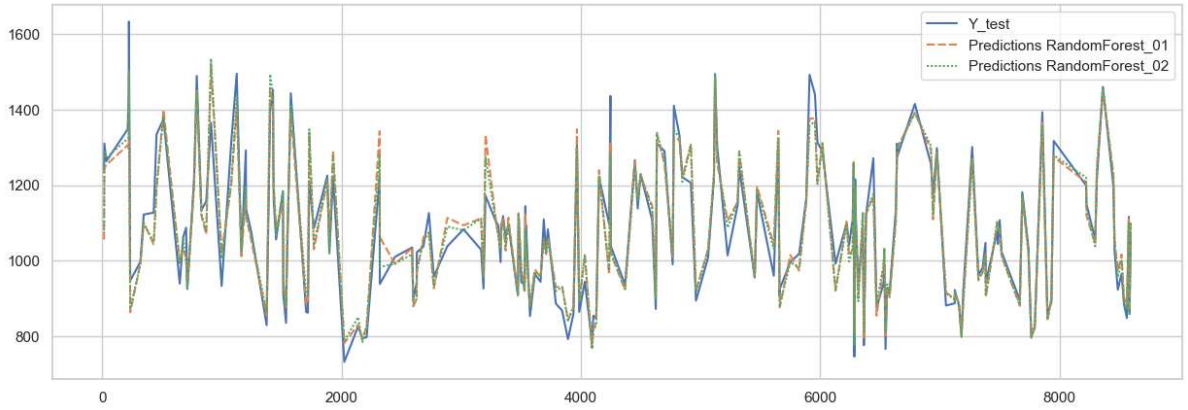
In [35]:
```
modelo_randomforest_2 = RandomForestRegressor()

modelo_randomforest_2.fit(X_2_train, Y_2_train)
```

Out[35]:
```
▾ RandomForestRegressor
RandomForestRegressor()
```

In [36]:
```
pred_randomforest_2 = modelo_randomforest_2.predict(X_2_test)

print('Regression Model without Seasons: R²={:.2f}'.format(metrics.r2_score(Y_test,
print('Regression Model with Seasons: R²={:.2f}'.format(metrics.r2_score(Y_2_test,
```

```
Regression Model without Seasons: R²=0.92
Regression Model with Seasons: R²=0.94
```

In [37]:
```
aux['Predictions RandomForest_02'] = pred_randomforest_2

plt.figure(figsize=(15,5))
sns.lineplot(data=aux.iloc[:200,:])
plt.show()
```



In [38]:
```
#It can be seen that the new model used the new information to react slightly bette
#in the predicted variable.
```

Data Preparation

Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

In [2]:
```
data = pd.read_csv('Datasets/heart.csv')
```

In [3]:
```
data.head()
```

Out[3]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

In [4]:
```
data.tail()
```

Out[4]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targe
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2	
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3	
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	2	
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2	
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3	

In [5]:
```
data.shape
```

Out[5]:
```
(1025, 14)
```

In [6]:
```
data.columns
```

Out[6]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

In [7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1025 non-null   int64  
 1   sex       1025 non-null   int64  
 2   cp        1025 non-null   int64  
 3   trestbps  1025 non-null   int64  
 4   chol      1025 non-null   int64  
 5   fbs       1025 non-null   int64  
 6   restecg   1025 non-null   int64  
 7   thalach   1025 non-null   int64  
 8   exang     1025 non-null   int64  
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64  
 11  ca        1025 non-null   int64  
 12  thal      1025 non-null   int64  
 13  target    1025 non-null   int64  
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

Data Preprocessing

In [8]:
```
data.describe()
```

Out[8]:

	age	sex	cp	trestbps	chol	fbs	restecg
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000	1025.000000
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.149268	0.529756
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.527878
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000	0.000000
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.000000	0.000000
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.00000	1.000000	2.000000

Checking for null value

In [9]:
```
data.isnull().sum()
```

Out[9]:
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

Checking duplicate value

In [10]:
```
# Check duplicate values
duplicate_rows = data[data.duplicated()]
duplicate_sort = duplicate_rows.sort_values(by=['age'])
print(duplicate_sort)
```

```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
118   29    1   1       130   204    0        0      202      0      0.0
64    29    1   1       130   204    0        0      202      0      0.0
668   29    1   1       130   204    0        0      202      0      0.0
15    34    0   1       118   210    0        1      192      0      0.7
779   34    0   1       118   210    0        1      192      0      0.7
..   ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
724   74    0   1       120   269    0        0      121      1      0.2
535   76    0   2       140   197    0        2      116      0      1.1
965   76    0   2       140   197    0        2      116      0      1.1
162   77    1   0       125   304    0        0      162      1      0.0
387   77    1   0       125   304    0        0      162      1      0.0

     slope  ca  thal  target
118      2   0     2       1
64       2   0     2       1
668      2   0     2       1
15       2   0     2       1
779      2   0     2       1
..     ...  ..   ...     ...
724      2   1     2       1
535      1   0     2       1
965      1   0     2       1
162      2   3     2       0
387      2   3     2       0

[723 rows x 14 columns]
```

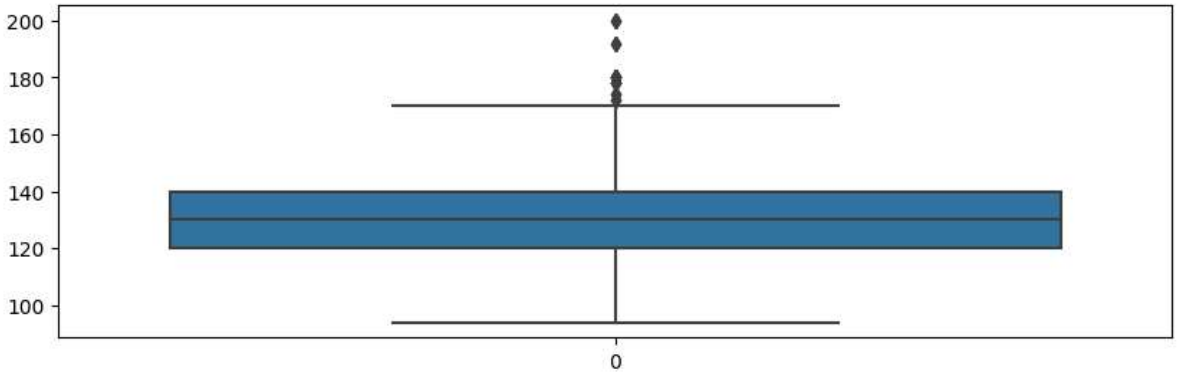
In [11]:
```
data.duplicated().sum()
```

Out[11]:
```
723
```

Checking outliers

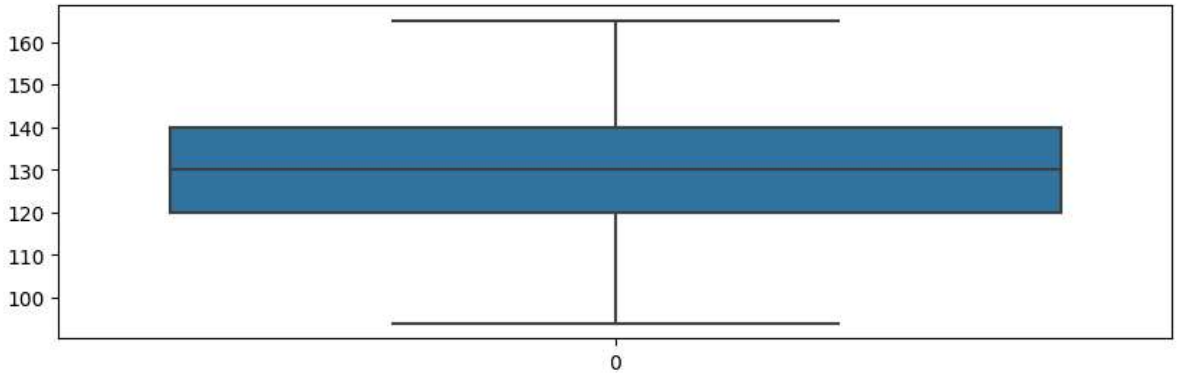
In [12]:
```
#trestbps outliers
plt.figure(figsize=(10,3))
sns.boxplot(data['trestbps'])
```

Out[12]:
```
<Axes: >
```



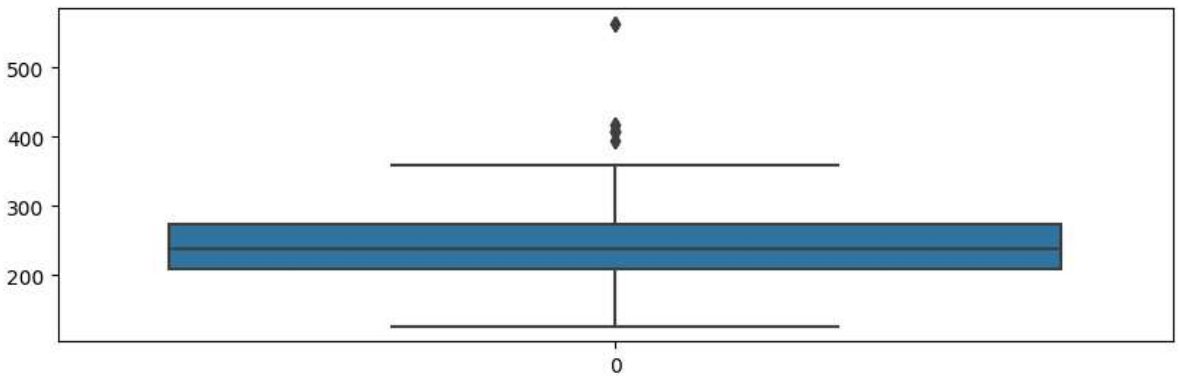
In [13]:
```
q1 = data['trestbps'].quantile(0.25)
q3 = data['trestbps'].quantile(0.75)
IQR = q3 - q1
lower_limit = q1 - 1.5 * IQR
upper_limit = q3 + 1.5 * IQR
data = data[(data['trestbps'] > lower_limit) & (data['trestbps'] < upper_limit)]
plt.figure(figsize=(10,3))
sns.boxplot(data['trestbps'])
```

Out[13]:
```
<Axes: >
```

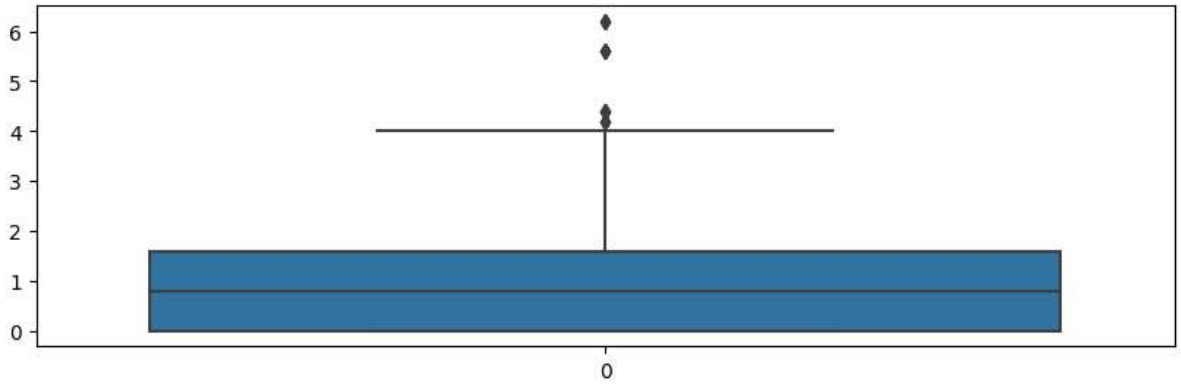


In [14]:
```
#chol
plt.figure(figsize=(10,3))
sns.boxplot(data['chol'])
```

Out[14]:
```
<Axes: >
```

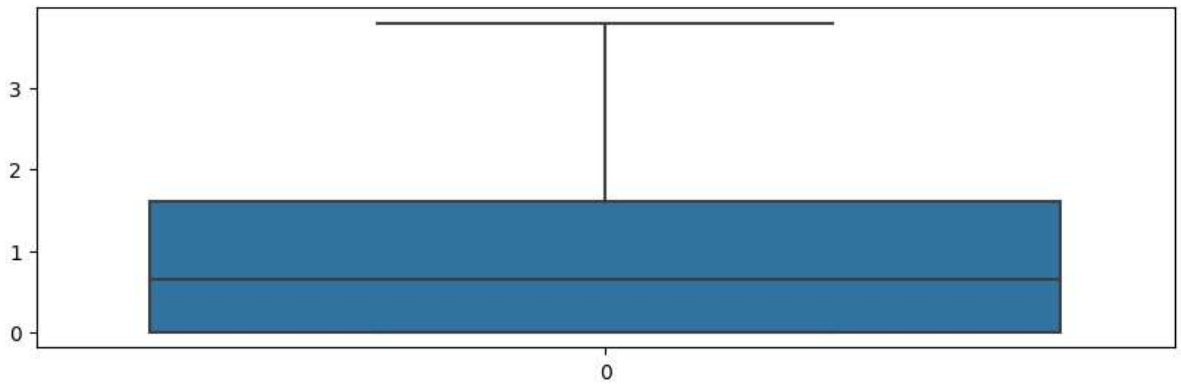


In [15]:
```
q1 = data['chol'].quantile(0.25)
q3 = data['chol'].quantile(0.75)
IQR = q3 - q1
lower_limit = q1 - 1.5 * IQR
upper_limit = q3 + 1.5 * IQR
data = data[(data['chol'] > lower_limit) & (data['chol'] < upper_limit)]
plt.figure(figsize=(10,3))
sns.boxplot(data['chol'])
```



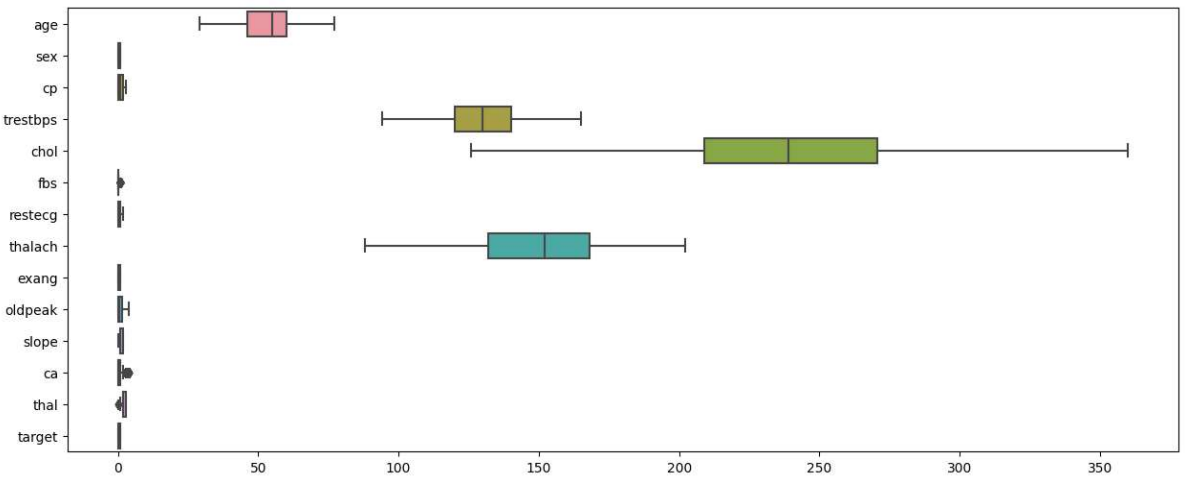
In [19]:
```
q1 = data['oldpeak'].quantile(0.25)
q3 = data['oldpeak'].quantile(0.75)
IQR = q3 - q1
lower_limit = q1 - 1.5 * IQR
upper_limit = q3 + 1.5 * IQR
data = data[(data['oldpeak'] > lower_limit) & (data['oldpeak'] < upper_limit)]
plt.figure(figsize=(10,3))
sns.boxplot(data['oldpeak'])
```

Out[19]:
```
<Axes: >
```



In [20]:
```
plt.figure(figsize=(15,6))
sns.boxplot(data=data, orient='h')
```

Out[20]:
```
<Axes: >
```



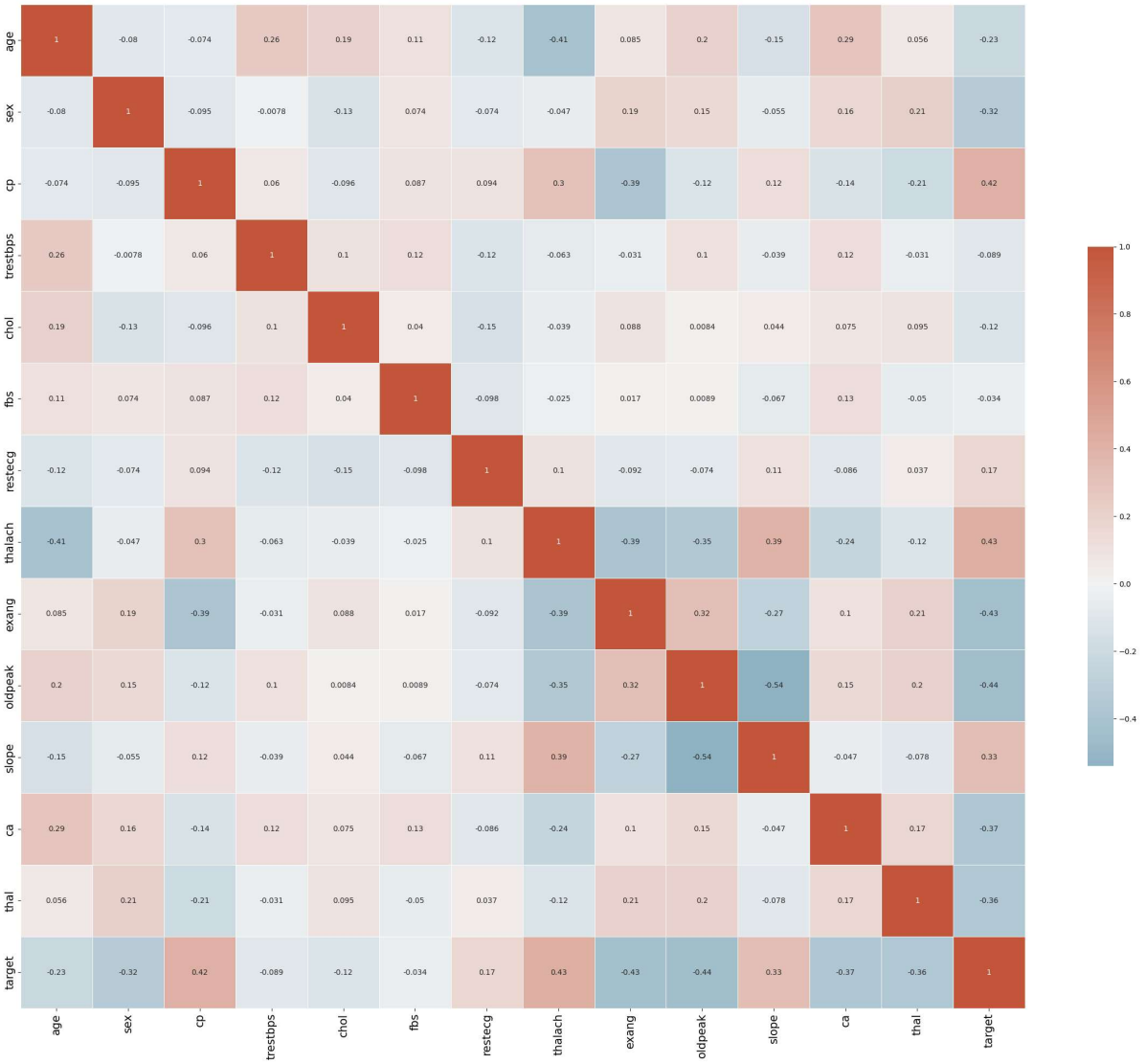
In [21]:
```
# Checking for balanced dataset
data['target'].value_counts()
```

Out[21]:
```
1    504
0    438
Name: target, dtype: int64
```

EDA

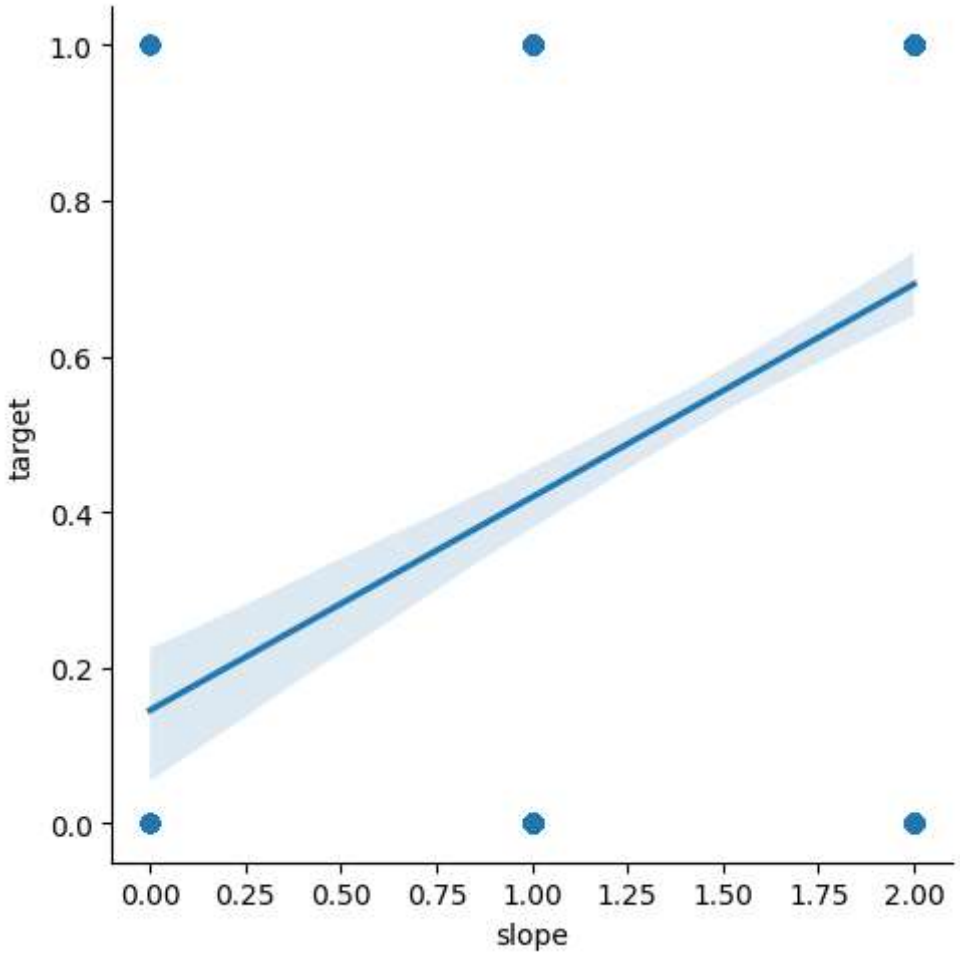
In [22]:
```
#Heatmap Correlation
import seaborn as sns

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(30, 25))

corr = data.corr()

matt = data.corr()
mask = np.zeros_like(corr, dtype=np.bool)
# Generate a custom diverging colormap
cmap = sns.diverging_palette(230,20, as_cmap=True)
# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(matt, mask=mask,cmap=cmap, vmax=1, center=0, annot =True,
            square=True, linewidths=.7, cbar_kws={"shrink": .5})
plt.xticks(rotation=90, fontsize=15)
plt.yticks(fontsize=15)
plt.show()
```

```
C:\Users\ASUS\AppData\Local\Temp\ipykernel_22252\3742216722.py:10: DeprecationWarn
ing: `np.bool` is a deprecated alias for the builtin `bool`. To silence this warni
ng, use `bool` by itself. Doing this will not modify any behavior and is safe. If
you specifically wanted the numpy scalar type, use `np.bool_` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdoc
s/release/1.20.0-notes.html#deprecations
  mask = np.zeros_like(corr, dtype=np.bool)
```



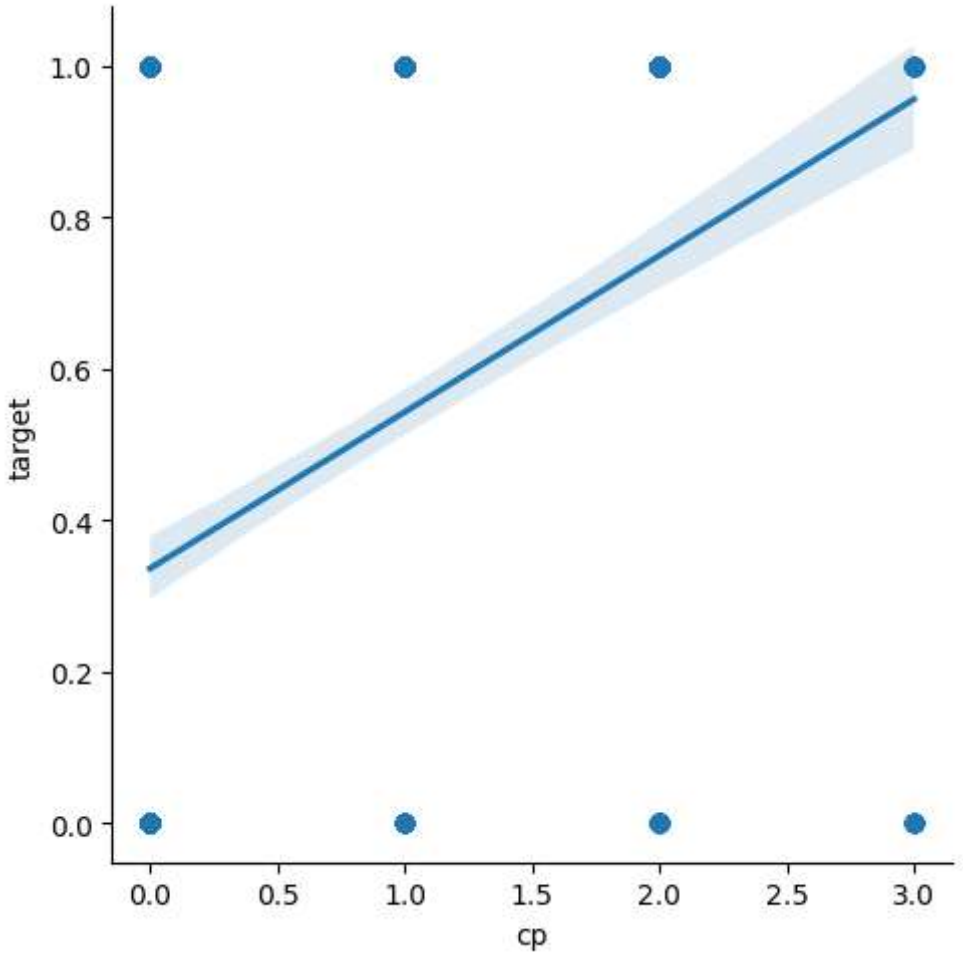
In [23]:
```
# slope vs target (plt.scatter(X,Y))
sns.lmplot(data=data, x="slope", y="target")
```

Out[23]:
```
<seaborn.axisgrid.FacetGrid at 0x1ffe63fa4d0>
```



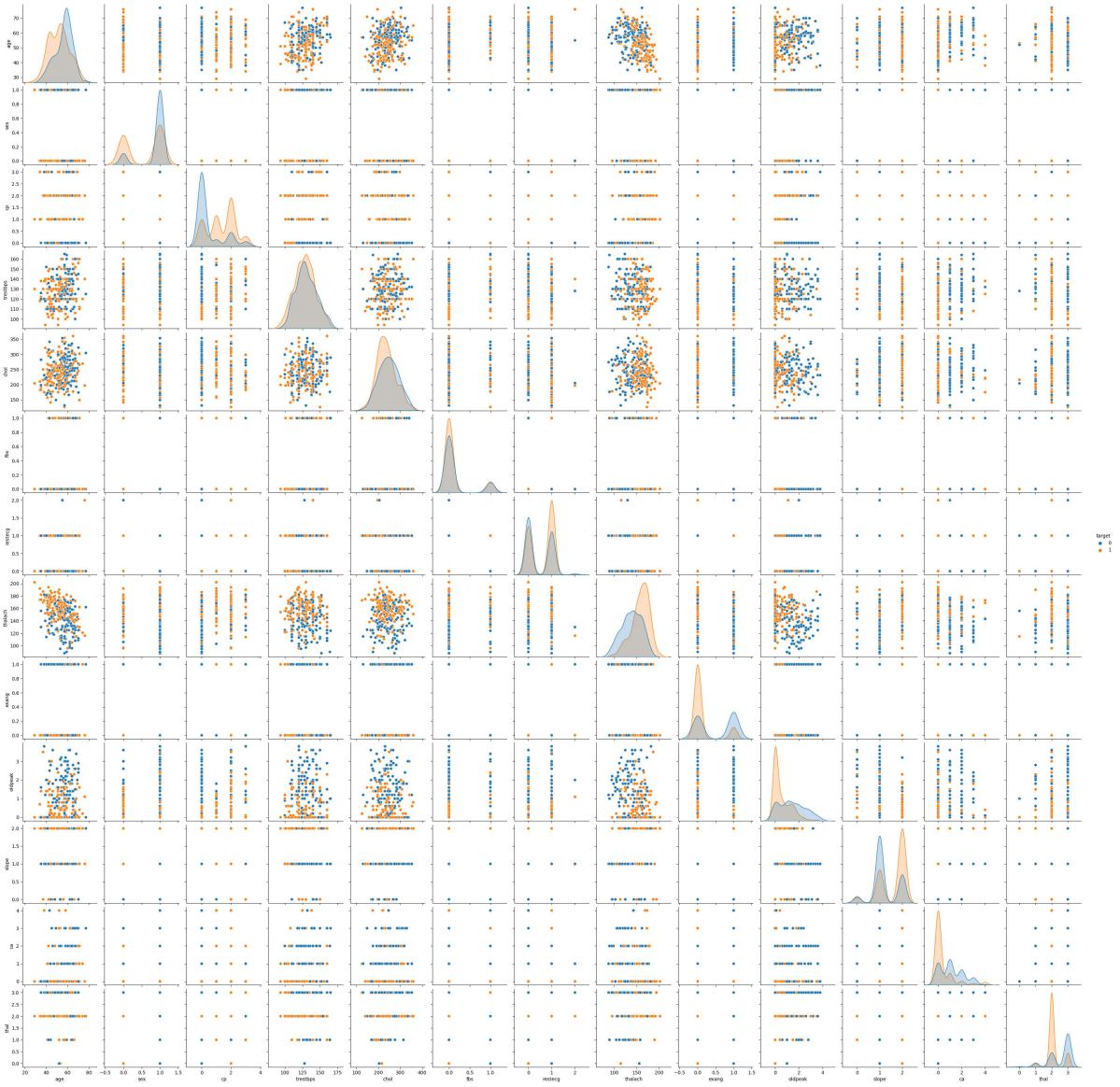
In [24]:
```
# thalach vs target (plt.scatter(X,Y))
sns.lmplot(data=data, x="thalach", y="target")
```

Out[24]:
```
<seaborn.axisgrid.FacetGrid at 0x1ffe84424d0>
```



In [27]:
```
sns.pairplot(data=data, hue="target")
```

Out[27]:
```
<seaborn.axisgrid.PairGrid at 0x1ffe8a3e350>
```



In [28]:
```
x=data.drop('target',axis=1)
x
```

Out[28]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	2
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3

942 rows × 13 columns

In [29]:
```
y=data['target']
y
```

Out[29]:
```
0       0
1       0
2       0
3       0
4       0
       ..
1020    1
1021    0
1022    0
1023    1
1024    0
Name: target, Length: 942, dtype: int64
```

In [ ]:
```

```

Univariate Selection For categorical Variable

In [30]:
```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
bestfeatures = SelectKBest(score_func=chi2)
```

In [31]:
```
fit = bestfeatures.fit(x,y)
scores = pd.DataFrame(fit.scores_)
```

In [32]:
```
dfcolumns = pd.DataFrame(x.columns)
```

In [33]:
```
featureScores = pd.concat([dfcolumns,scores],axis=1)
featureScores.columns = ['Label','Score']
```

In [34]:
```
featureScores.sort_values(by='Score',ascending=False)
```

Out[34]:

	Label	Score
7	thalach	601.806705
9	oldpeak	193.196121
11	ca	180.353925
2	cp	179.716870
8	exang	117.508421
4	chol	115.677722
0	age	75.723136
1	sex	28.526796
10	slope	25.345451
12	thal	19.464820
6	restecg	12.709173
3	trestbps	12.349684
5	fbs	0.963264

Feature Engineering

In [35]:
```
new_sex=pd.get_dummies(data=data['sex'],prefix='sex')
new_sex
```

Out[35]:

	sex_0	sex_1
0	0	1
1	0	1
2	0	1
3	0	1
4	1	0
...	...	...
1020	0	1
1021	0	1
1022	0	1
1023	1	0
1024	0	1

942 rows × 2 columns

In [36]:
```
new_cp=pd.get_dummies(data['cp'],prefix='chestPain')
new_cp
```

Out[36]:

	chestPain_0	chestPain_1	chestPain_2	chestPain_3
0	1	0	0	0
1	1	0	0	0
2	1	0	0	0
3	1	0	0	0
4	1	0	0	0
...	...	...	...	...
1020	0	1	0	0
1021	1	0	0	0
1022	1	0	0	0
1023	1	0	0	0
1024	1	0	0	0

942 rows × 4 columns

In [37]:
```
new_exang=pd.get_dummies(data['exang'],prefix='exang')
new_exang
```

Out[37]:

	exang_0	exang_1
0	1	0
1	0	1
2	0	1
3	1	0
4	1	0
...	...	...
1020	0	1
1021	0	1
1022	0	1
1023	1	0
1024	1	0

942 rows × 2 columns

In [38]:
```
new_slope=pd.get_dummies(data['slope'],prefix='slope')
new_slope
```

Out[38]:

	slope_0	slope_1	slope_2
0	0	0	1
1	1	0	0
2	1	0	0
3	0	0	1
4	0	1	0
...	...	...	...
1020	0	0	1
1021	0	1	0
1022	0	1	0
1023	0	0	1
1024	0	1	0

942 rows × 3 columns

In [39]:
```
new_thal=pd.get_dummies(data['thal'],prefix='thal')
new_thal
```

Out[39]:

	thal_0	thal_1	thal_2	thal_3
0	0	0	0	1
1	0	0	0	1
2	0	0	0	1
3	0	0	0	1
4	0	0	1	0
...	...	...	...	...
1020	0	0	1	0
1021	0	0	0	1
1022	0	0	1	0
1023	0	0	1	0
1024	0	0	0	1

942 rows × 4 columns

In [40]:
```
new_ca=pd.get_dummies(data['ca'],prefix='ca')
new_ca
```

Out[40]:

	ca_0	ca_1	ca_2	ca_3	ca_4
0	0	0	1	0	0
1	1	0	0	0	0
2	1	0	0	0	0
3	0	1	0	0	0
4	0	0	0	1	0
...	...	...	...	...	...
1020	1	0	0	0	0
1021	0	1	0	0	0
1022	0	1	0	0	0
1023	1	0	0	0	0
1024	0	1	0	0	0

942 rows × 5 columns

In [41]:
```
app=[data,new_sex,new_cp,new_ca,new_thal,new_exang,new_slope]
```

In [42]:
```
df1=pd.concat(app,axis=1)
```

In [43]:
```
df1.head()
```

Out[43]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	...	ca_4	thal_0	thal_1
0	52	1	0	125	212	0	1	168	0	1.0	...	0	0	0
1	53	1	0	140	203	1	0	155	1	3.1	...	0	0	0
2	70	1	0	145	174	0	1	125	1	2.6	...	0	0	0
3	61	1	0	148	203	0	1	161	0	0.0	...	0	0	0
4	62	0	0	138	294	1	1	106	0	1.9	...	0	0	0

5 rows × 34 columns

In [44]:
```
df1.drop(['sex','cp','thal','exang','ca','slope'],axis=1,inplace=True)
```

Feature Scalling

In [45]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
```

In [46]:
```
df1[['age','trestbps','chol','oldpeak','thalach']]=sc.fit_transform(df1[['age','tr
```

In [47]:
```
X=df1.drop('target',axis=1)
X
```

Out[47]:

	age	trestbps	chol	fbs	restecg	thalach	oldpeak	sex_0	sex_1	chestPain_
0	-0.210426	-0.290753	-0.664118	0	1	0.803573	0.046685	0	1	
1	-0.101573	0.736589	-0.864245	1	0	0.235871	2.122016	0	1	
2	1.748925	1.079036	-1.509101	0	1	-1.074213	1.627890	0	1	
3	0.769250	1.284505	-0.864245	0	1	0.497887	-0.941568	0	1	
4	0.878103	0.599610	1.159267	1	1	-1.903932	0.936113	1	0	
...	...	...	...	...	...	...	...	...	...	
1020	0.551544	0.736589	-0.463990	0	1	0.628896	-0.941568	0	1	
1021	0.660397	-0.290753	0.358757	0	0	-0.375502	1.825540	0	1	
1022	-0.754690	-1.318095	0.736775	0	0	-1.379899	0.046685	0	1	
1023	-0.428131	-1.318095	0.269811	0	0	0.410548	-0.941568	1	0	
1024	0.007280	-0.633200	-1.197791	0	1	-1.598246	0.441986	0	1	

942 rows × 27 columns

In [48]:
```
Y=df1['target']
Y
```

Out[48]:
```
0       0
1       0
2       0
3       0
4       0
       ..
1020    1
1021    0
1022    0
1023    1
1024    0
Name: target, Length: 942, dtype: int64
```

In [49]:
```
scaler = StandardScaler()
scaler.fit(X)
X_scaled = scaler.transform(X)
```

In [50]:
```
from sklearn.model_selection import train_test_split

#X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, random_state=0)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size = 0.2,


print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(753, 27) (189, 27) (753,) (189,)
```

Cross Validation

In [51]:
```
#Logistic Regression

from sklearn.datasets import load_iris
from sklearn.model_selection import cross_val_score,KFold
from sklearn.linear_model import LogisticRegression
iris=load_iris()
X=X_train
Y=y_train
logreg=LogisticRegression()
kf=KFold(n_splits=5)
score=cross_val_score(logreg,X,Y,cv=kf)
print("Cross Validation Scores are {}".format(score))
print("Average Cross Validation score :{}".format(score.mean()))
```

```
Cross Validation Scores are [0.8807947  0.82781457 0.89403974 0.82       0.8866666
7]
Average Cross Validation score :0.8618631346578365
```

In [52]:
```
#DecisionTree

from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import KFold, cross_val_score

#X, y = datasets.load_iris(return_X_y=True)

clf = DecisionTreeClassifier(random_state=42)

k_folds = KFold(n_splits = 5)

scores = cross_val_score(clf, X_train, y_train, cv = k_folds)

print("Cross Validation Scores: ", scores)
```

```
print("Average CV Score: ", scores.mean())
print("Number of CV Scores used in Average: ", len(scores))
```

```
Cross Validation Scores:  [1.         0.98675497 0.97350993 0.99333333 0.97333333]
Average CV Score:  0.9853863134657835
Number of CV Scores used in Average:  5
```

Modelling

In [53]:
```
#Logistic Regression
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression()
logreg.fit(X_train,y_train)

print("Logistic Regression train score : {:.4f}".format(logreg.score(X_train, y_tr
print("Logistic Regression test score : {:.4f}".format(logreg.score(X_test, y_test
```

```
Logistic Regression train score : 0.8792
Logistic Regression test score : 0.8624
```

In [54]:
```
#Decision Tree
from sklearn.tree import DecisionTreeClassifier

tree = DecisionTreeClassifier(max_depth=8, random_state=0)
tree.fit(X_train, y_train)

print("Decision Tree train score : {:.4f}".format(tree.score(X_train, y_train)))
print("Decision Tree test score : {:.4f}".format(tree.score(X_test, y_test)))
```

```
Decision Tree train score : 1.0000
Decision Tree test score : 1.0000
```

Model Evaluation

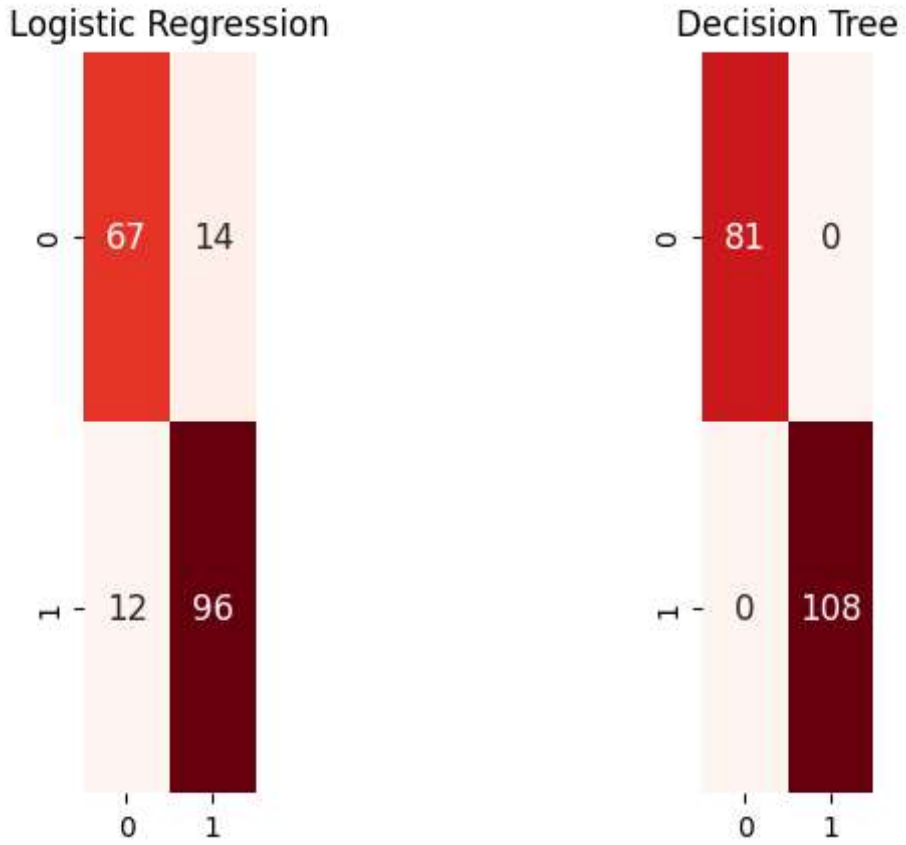
In [55]:
```
print("Logistic Regression test score : {:.4f}".format(logreg.score(X_test, y_test
print("Decision Tree test score {:.4f}".format(tree.score(X_test, y_test)))
```

```
Logistic Regression test score : 0.8624
Decision Tree test score 1.0000
```

In [56]:
```
pred_logreg = logreg.predict(X_test)
pred_tree = tree.predict(X_test)
```

In [57]:
```
from sklearn.metrics import confusion_matrix

cm_logreg = confusion_matrix(y_test, pred_logreg)
cm_tree = confusion_matrix(y_test, pred_tree)
```

In [58]:
```
plt.subplot(1,5,2)
plt.title("Logistic Regression")
sns.heatmap(cm_logreg, annot=True, cmap="Reds", fmt="d", cbar=False, annot_kws={"s

plt.subplot(1,5,5)
plt.title("Decision Tree")
sns.heatmap(cm_tree, annot=True, cmap="Reds", fmt="d", cbar=False, annot_kws={"siz

plt.show()
```



In [59]:
```
from sklearn.metrics import classification_report

print('---------------------------------------------------------------\n\nLogis
print('---------------------------------------------------------------')
print(classification_report(y_test, pred_logreg, target_names=["Have Disease", "Do
print('---------------------------------------------------------------\n\nDecis
print('---------------------------------------------------------------')
print(classification_report(y_test, pred_tree, target_names=["Have Disease", "Don'
```

```
---------------------------------------------------------------

Logistic Regression
---------------------------------------------------------------
                    precision    recall  f1-score   support

      Have Disease       0.85      0.83      0.84        81
Don't have Disease       0.87      0.89      0.88       108

          accuracy                           0.86       189
         macro avg       0.86      0.86      0.86       189
      weighted avg       0.86      0.86      0.86       189

---------------------------------------------------------------

Decision Tree
---------------------------------------------------------------
                    precision    recall  f1-score   support

      Have Disease       1.00      1.00      1.00        81
Don't have Disease       1.00      1.00      1.00       108

          accuracy                           1.00       189
         macro avg       1.00      1.00      1.00       189
      weighted avg       1.00      1.00      1.00       189
```

ASSIGNMENT NO: 7

Title:

Visualize the data using Python libraries matplotlib, seaborn by plotting the graphs**Problem**

Theory:

1. Matplotlib: Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It provides a wide range of plotting functions and features, allowing users to create line plots, scatter plots, bar plots, histograms, 3D plots, and much more. Matplotlib provides a highly customizable interface, giving users control over various aspects of their plots, such as colors, labels, annotations, and axes. It is widely used in scientific computing, data analysis, and data visualization domains.

2. Seaborn: Seaborn is a Python data visualization library built on top of Matplotlib. It provides a high-level interface and simplifies the creation of attractive statistical graphics. Seaborn enhances Matplotlib's capabilities by providing additional plot types and themes, making it easier to create visually appealing plots with fewer lines of code. Seaborn includes functions for creating statistical visualizations like scatter plots, box plots, violin plots, heatmaps, and regression plots. It also provides tools for handling categorical data and integrating with Pandas data structures.

Both Matplotlib and Seaborn are widely used in the data science community and can be used together to create rich and informative visualizations. They offer a variety of customization options and support for different plot types, making them valuable tools for exploring and communicating data.

Code & output:

In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
dataset1 = pd.read_csv("Datasets/heart.csv")
dataset2 = pd.read_csv("Datasets/AirQuality.csv", sep=";")
```

In [18]:
```
dataset1.head()
```

Out[18]:

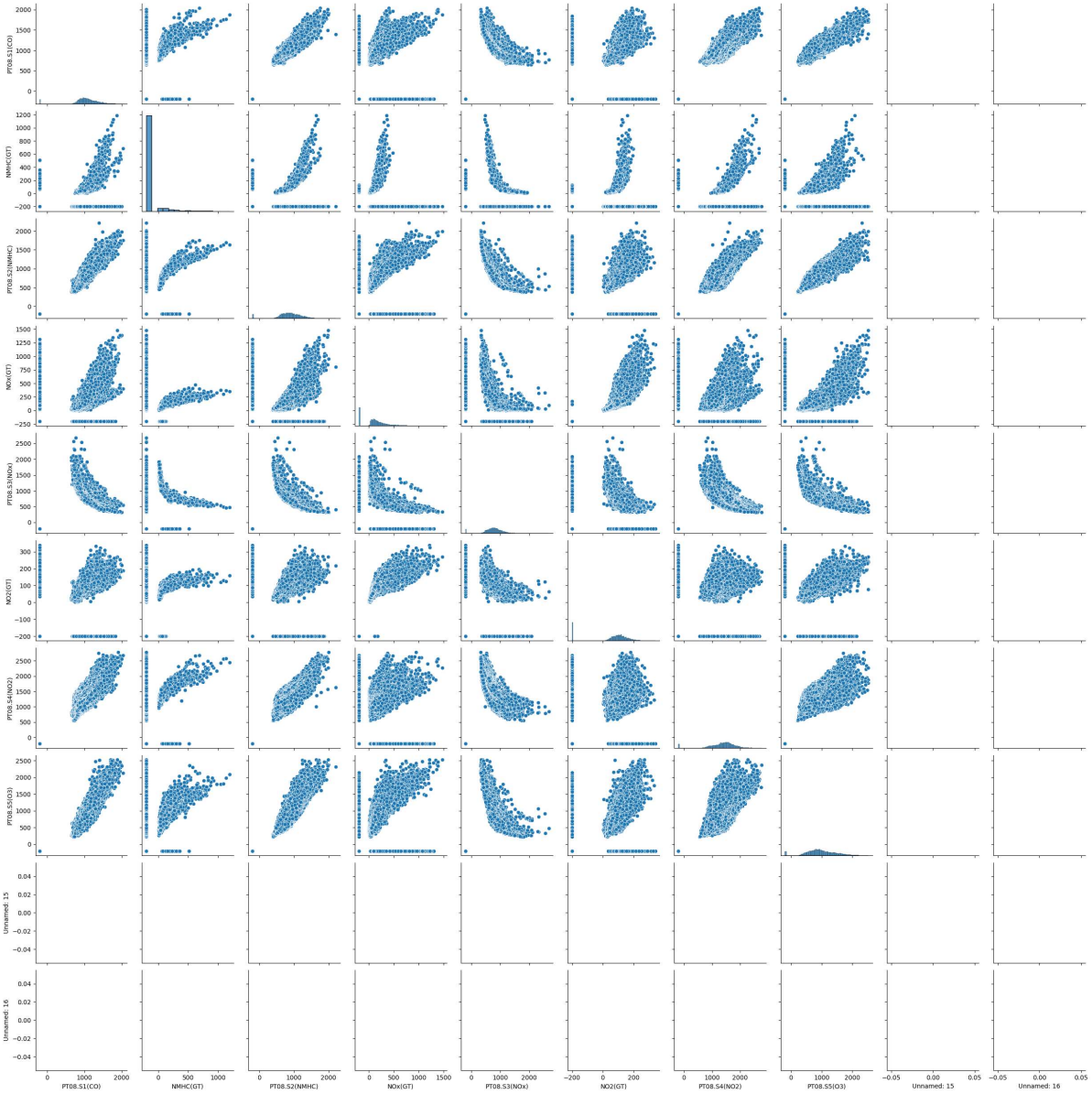
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

In [19]:
```
dataset2.head()
```

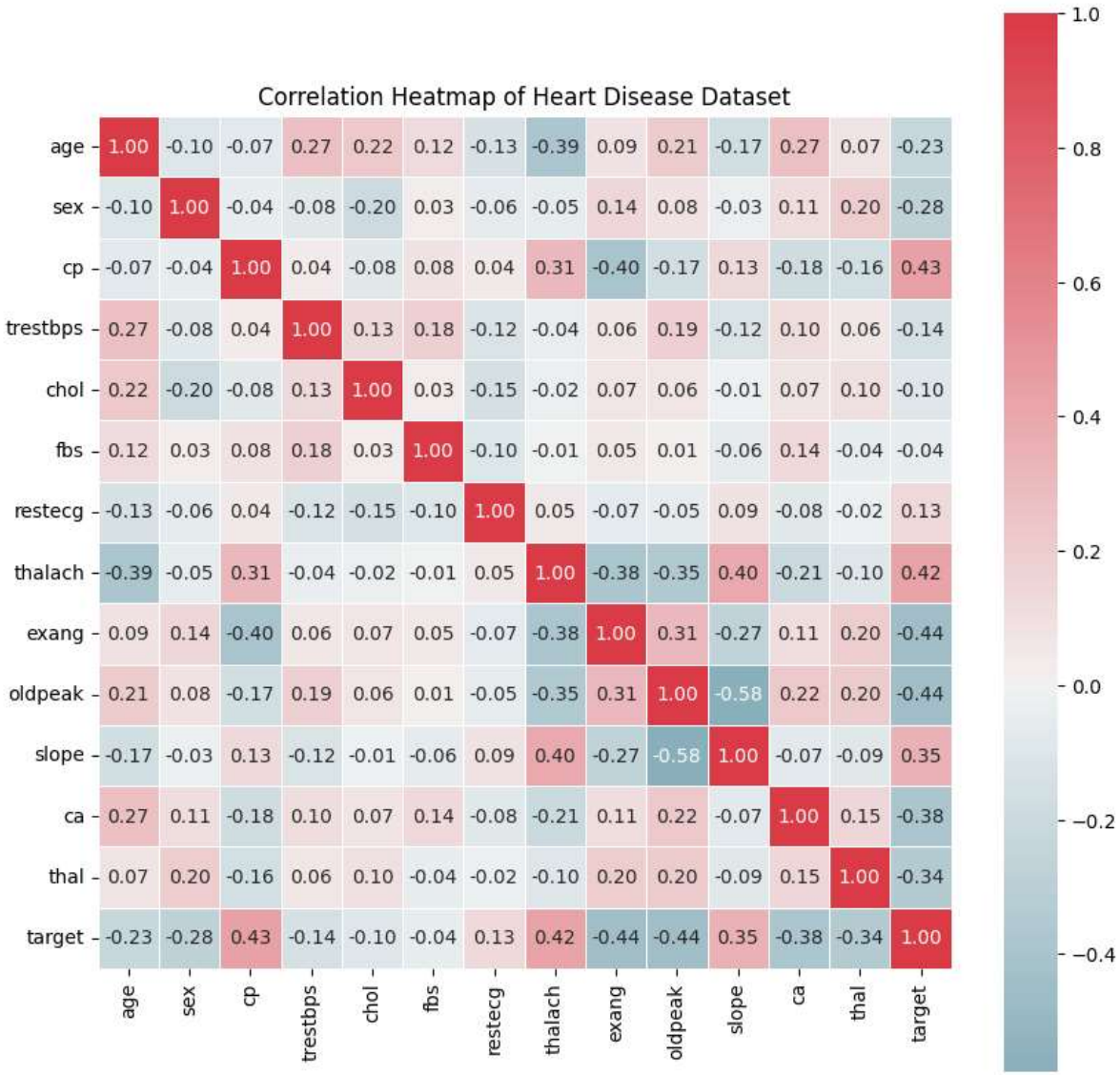
Out[19]:

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)
0	10/03/2004	18.00.00	2,6	1360.0	150.0	11,9	1046.0	166.0
1	10/03/2004	19.00.00	2	1292.0	112.0	9,4	955.0	103.0
2	10/03/2004	20.00.00	2,2	1402.0	88.0	9,0	939.0	131.0
3	10/03/2004	21.00.00	2,2	1376.0	80.0	9,2	948.0	172.0
4	10/03/2004	22.00.00	1,6	1272.0	51.0	6,5	836.0	131.0

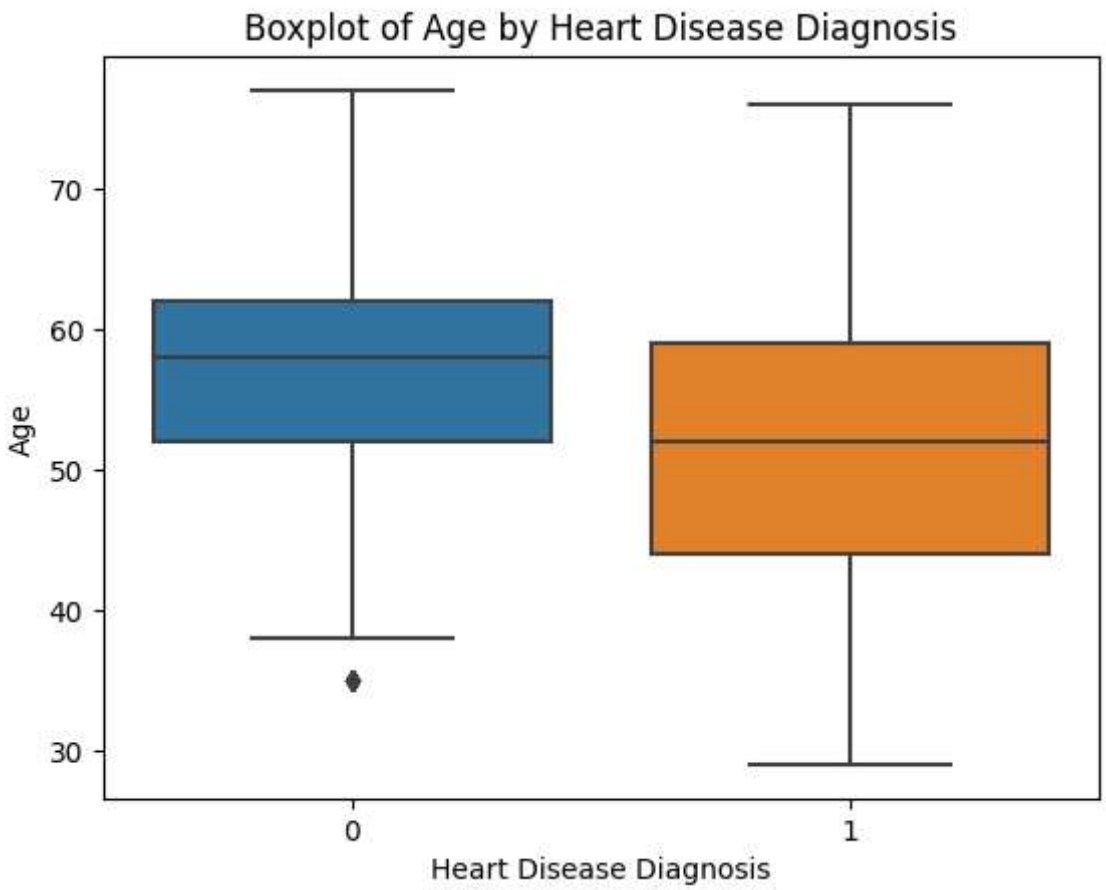
In [4]:
```
sns.pairplot(dataset2)
plt.show()
```



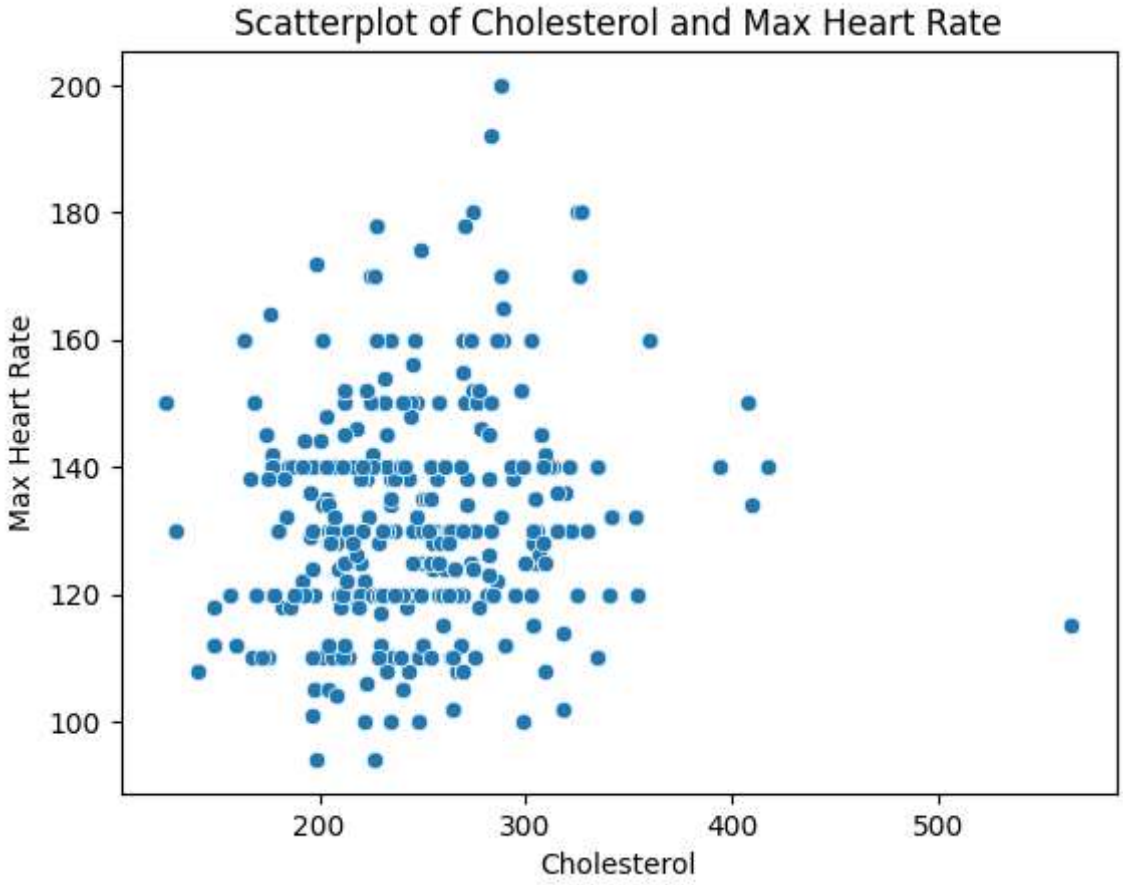
In [5]:
```
corr = dataset1.corr() # Compute the correlation matrix

fig, ax = plt.subplots(figsize=(10, 10)) # Set up the matplotlib figure

cmap = sns.diverging_palette(220, 10, as_cmap=True) # Define a custom colormap

sns.heatmap(corr, cmap=cmap, center=0, annot=True, fmt=".2f", square=True, linewid

# Draw the heatmap with the mask and aspect ratio
plt.yticks(rotation=0) # Set the y-axis label to be horizontal
plt.title("Correlation Heatmap of Heart Disease Dataset") # Set the title of the p
plt.show() # Show the plot
```



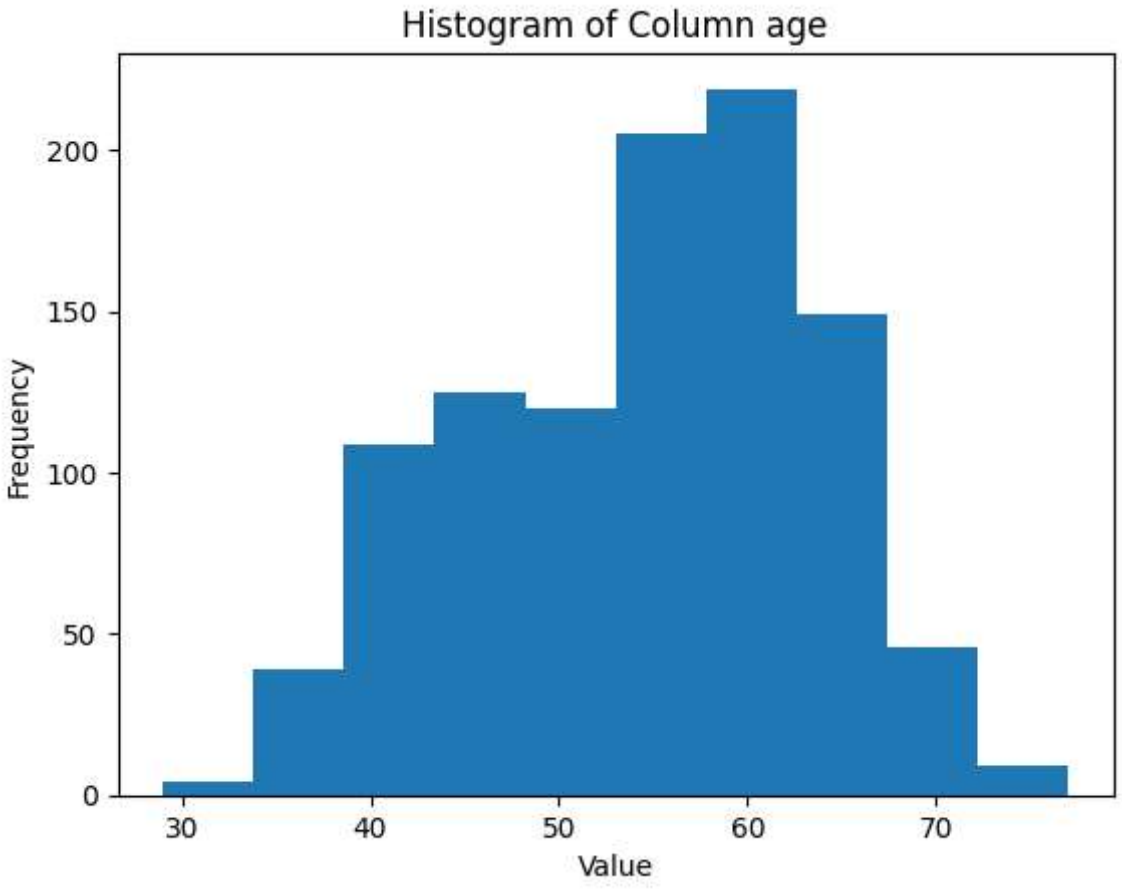
In [9]:
```
sns.boxplot(x='target', y='age', data=dataset1)
plt.title('Boxplot of Age by Heart Disease Diagnosis')
plt.xlabel('Heart Disease Diagnosis')
plt.ylabel('Age')
plt.show()
```



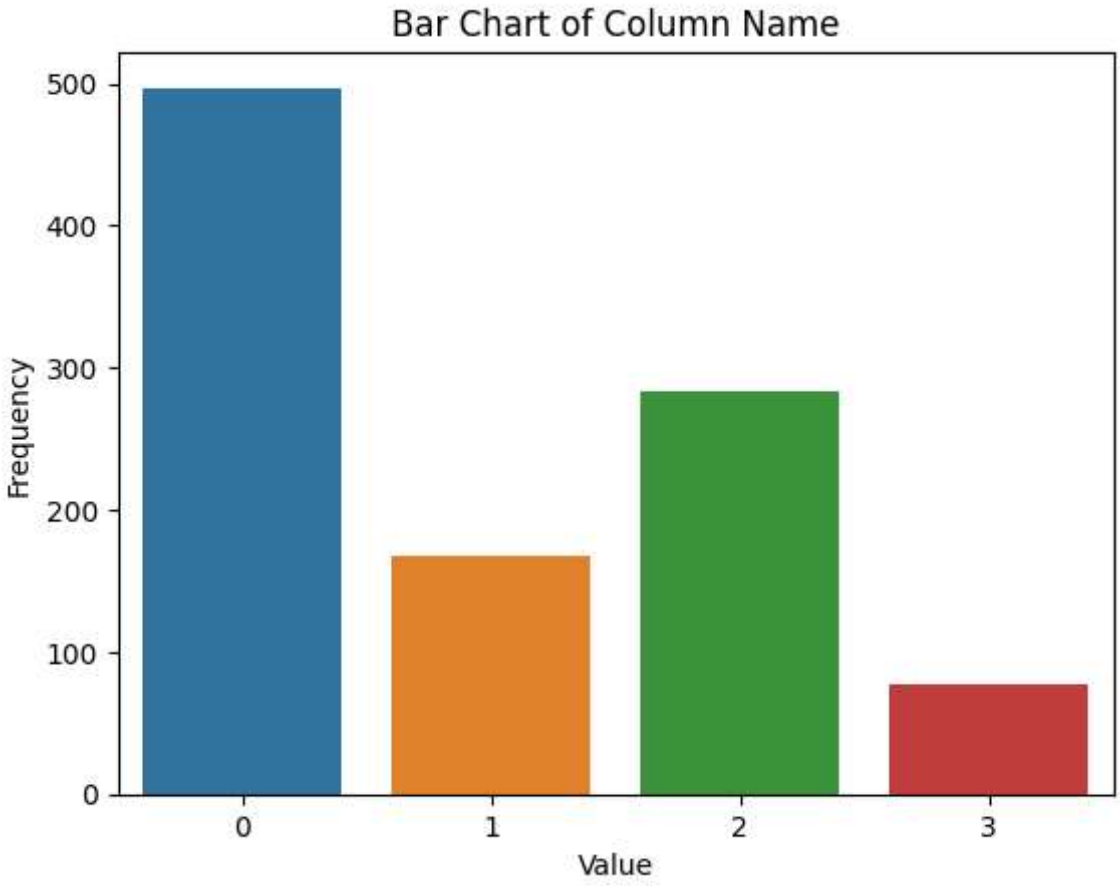
In [22]:
```
sns.scatterplot(x='chol', y='trestbps', data=dataset1)
plt.title('Scatterplot of Cholesterol and Max Heart Rate')
plt.xlabel('Cholesterol')
plt.ylabel('Max Heart Rate')
plt.show()
```



In [12]:
```
plt.hist(dataset1['age'], bins=10)
plt.title('Histogram of Column age')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```

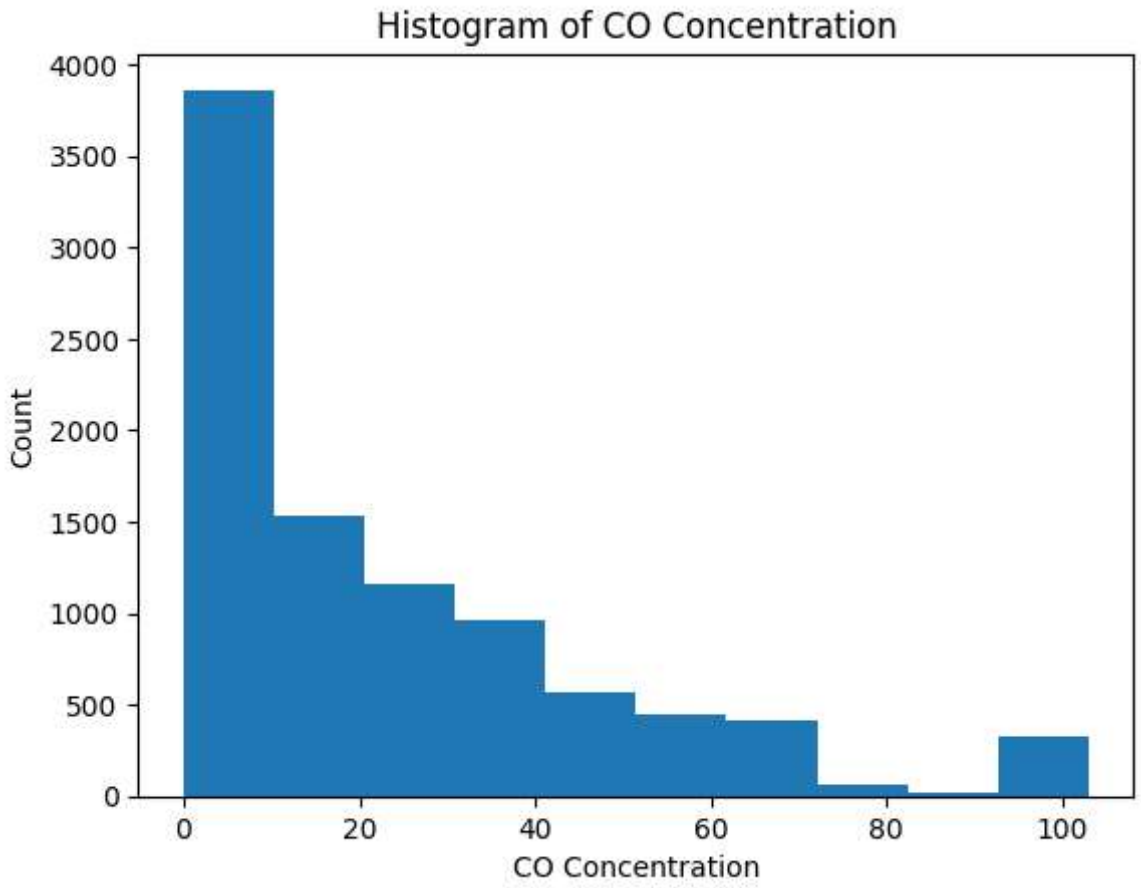


In [13]:
```
sns.countplot(x='cp', data=dataset1)
plt.title('Bar Chart of Column Name')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```

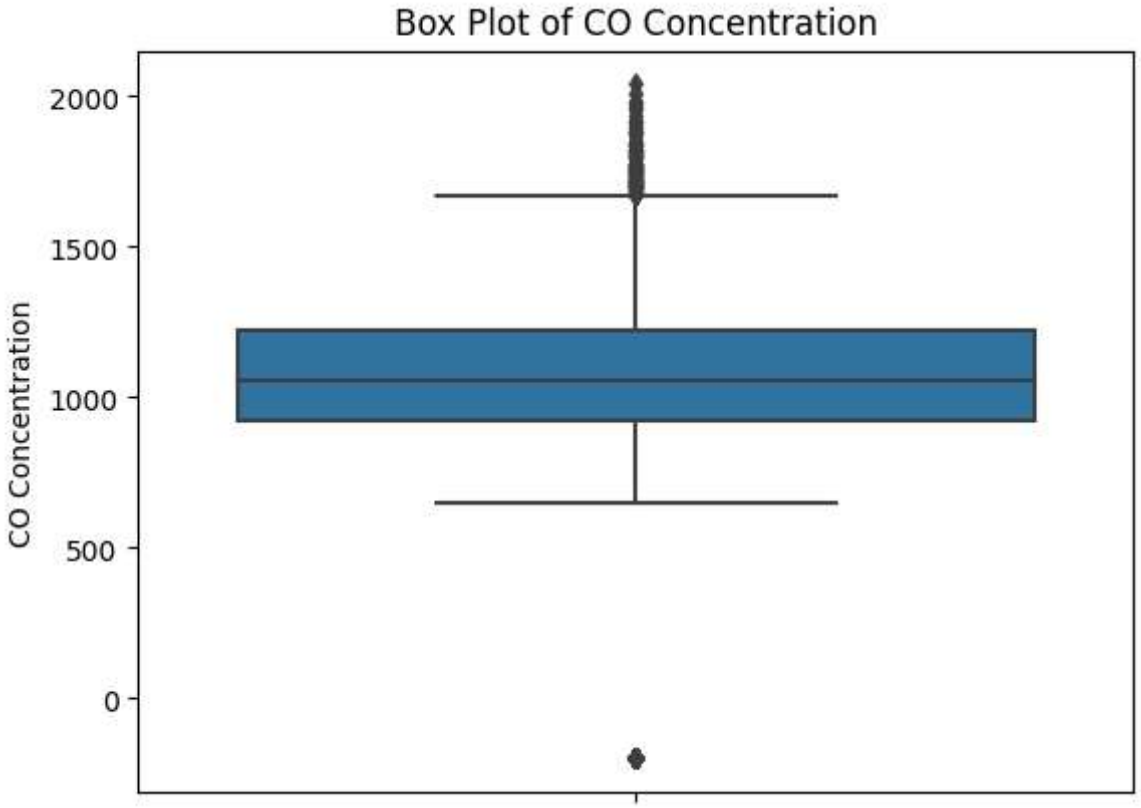


Air Quality

In [14]:
```
plt.hist(dataset2['CO(GT)'].dropna(), bins=10)
plt.xlabel('CO Concentration')
plt.ylabel('Count')
plt.title('Histogram of CO Concentration')
plt.xticks([0, 20, 40, 60, 80, 100], ['0', '20', '40', '60', '80', '100'])
plt.show()
```

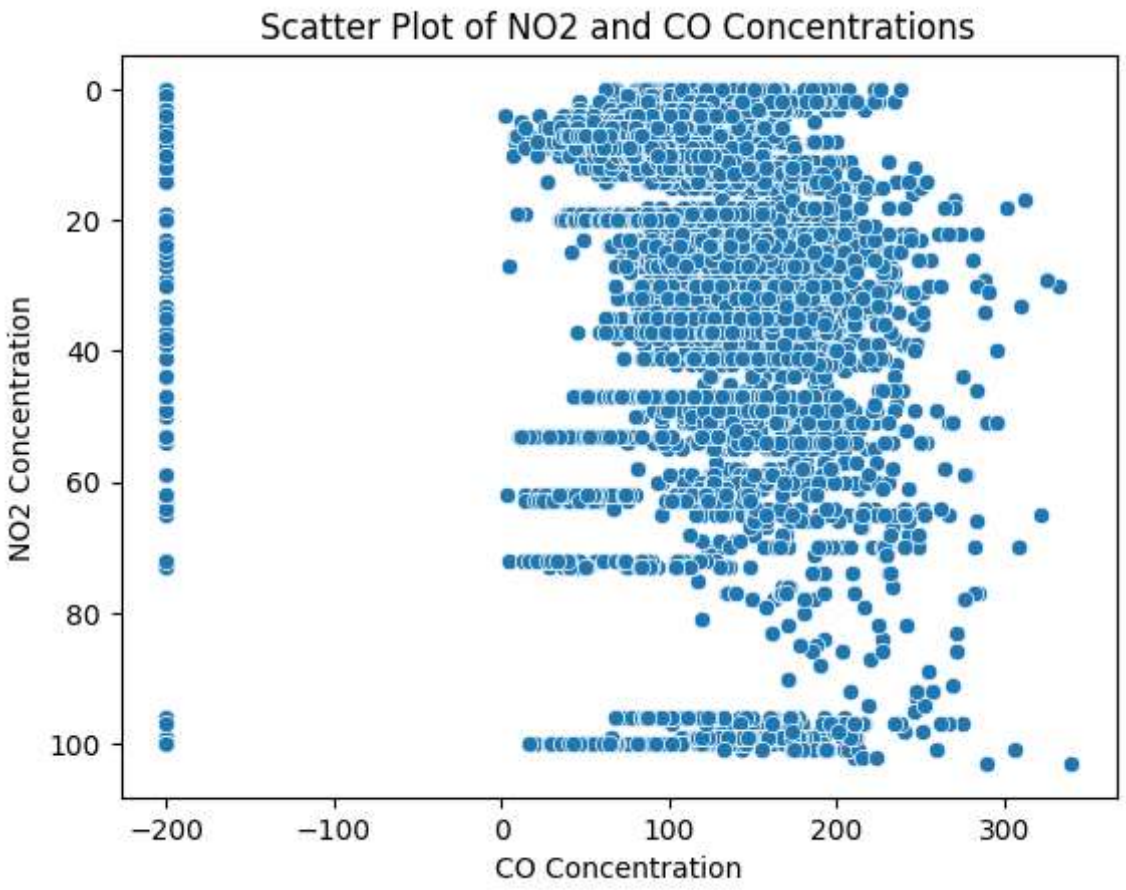


In [15]:
```
sns.boxplot(y=dataset2['PT08.S1(CO)'].dropna().astype(float))
plt.ylabel('CO Concentration')
plt.title('Box Plot of CO Concentration')
plt.show()
```

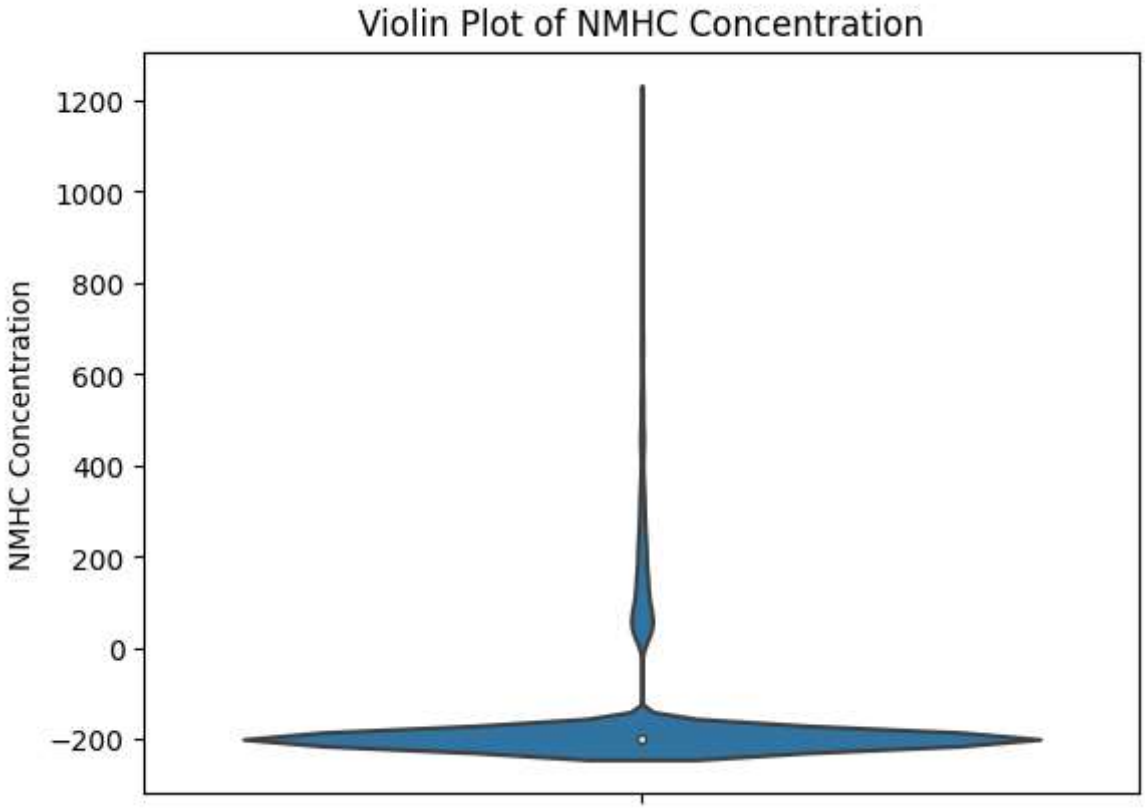


In [16]:
```
sns.scatterplot(x='NO2(GT)', y='CO(GT)', data=dataset2.dropna(subset=['NO2(GT)', '
plt.xlabel('CO Concentration')
plt.ylabel('NO2 Concentration')
```

```
plt.title('Scatter Plot of NO2 and CO Concentrations')
plt.yticks([0, 20, 40, 60, 80, 100], ['0', '20', '40', '60', '80', '100'])
plt.show()
```



In [17]:
```
sns.violinplot(y='NMHC(GT)', data=dataset2.dropna(subset=['NMHC(GT)']))
plt.ylabel('NMHC Concentration')
plt.title('Violin Plot of NMHC Concentration')
plt.show()
```



ASSIGNMENT NO: 8

Title:

Perform the following data visualization operations using Tableau on Adult and Iris datasets.
a. 1D (Linear) Data visualization

b. 2D (Planar) Data Visualization

c. 3D (Volumetric) Data Visualization

d. Temporal Data Visualization

e. Multidimensional Data Visualization

f. Tree/ Hierarchical Data visualization

g. Network Data visualization

Problem Statement:

Perform visualization using Tableau.

Theory:

Data visualization or data visualization is viewed by many disciplines as a modern equivalent of visual communication. It involves the creation and study of the visual representation of data, meaning "information that has been abstracted in some schematic form, including attributes or variables for the units of information".

Data visualization refers to the techniques used to communicate data or information by encoding it as visual objects (e.g., points, lines or bars) contained in graphics. The goal is to communicate information clearly and efficiently to users. It is one of the steps in data analysis or data science

3D/Volumetric

Broadly, examples of scientific visualization: 3D computer models

In 3D computer graphics, 3D modeling (or three-dimensional modeling) is the process of developing a mathematical representation of any surface of an object (either inanimate or living) in three dimensions via specialized software. The product is called a 3D model. Someone who works with 3D models may be referred to as a 3D artist. It can be displayed as a two-dimensional image through a process called 3D rendering or used in a computer simulation of physical phenomena. The model can also be physically created using 3D printing devices.

surface and volume rendering

Rendering is the process of generating an image from a model, by means of computer programs. The model is a description of three-dimensional objects in a strictly defined language or data structure. It would contain geometry, viewpoint, texture, lighting, and shading information. The image is a digital image or raster graphics image. The term

may be by analogy with an "artist's rendering" of a scene. 'Rendering' is also used to describe the process of calculating effects in a video editing file to produce final video output.

Volume rendering is a technique used to display a 2D projection of a 3D discretely sampled data set. A typical 3D data set is a group of 2D slice images acquired by a CT or MRI scanner. Usually these are acquired in a regular pattern (e.g., one slice every millimeter) and usually have a regular number of image pixels in a regular pattern. This is

an example of a regular volumetric grid, with each volume element, or voxel represented by a single value that is obtained by sampling the immediate area surrounding the voxel.

computer simulations

Computer simulation is a computer program, or network of computers, that attempts to simulate an abstract model of a particular system. Computer simulations have become a useful part of mathematical modeling of many natural systems in

physics, and computational physics, chemistry and biology; human systems in economics, psychology, and social science; and in the process of engineering and new technology, to gain insight into the operation of those systems, or to observe their behavior.[6] The simultaneous visualization and simulation of a system is called visulation.

Examples:
Temporal

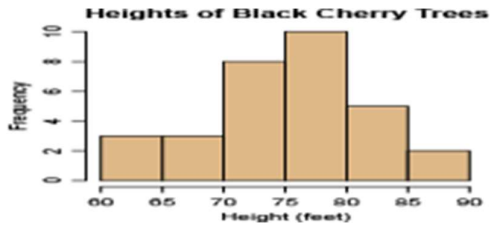
timeline

Tools: SIMILE Timeline, TimeFlow, Timeline JS,

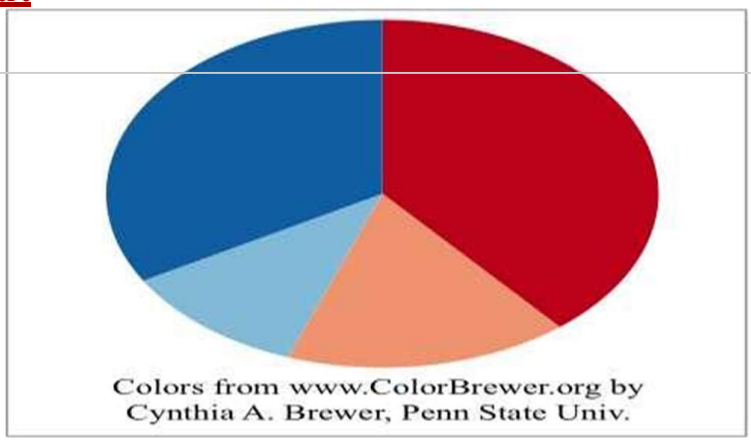
ExcelImage:
Friendly, M. & Denis, D. J. (2001). Milestones in the history of thematic cartography, statistical graphics, and data visualization. Web
document, http://www.datavis.ca/milestones/. Accessed: August 30, 2012.

time series

histogram



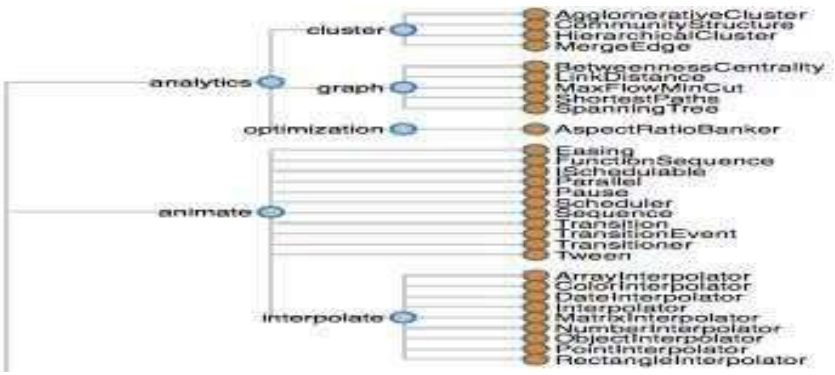
pie chart



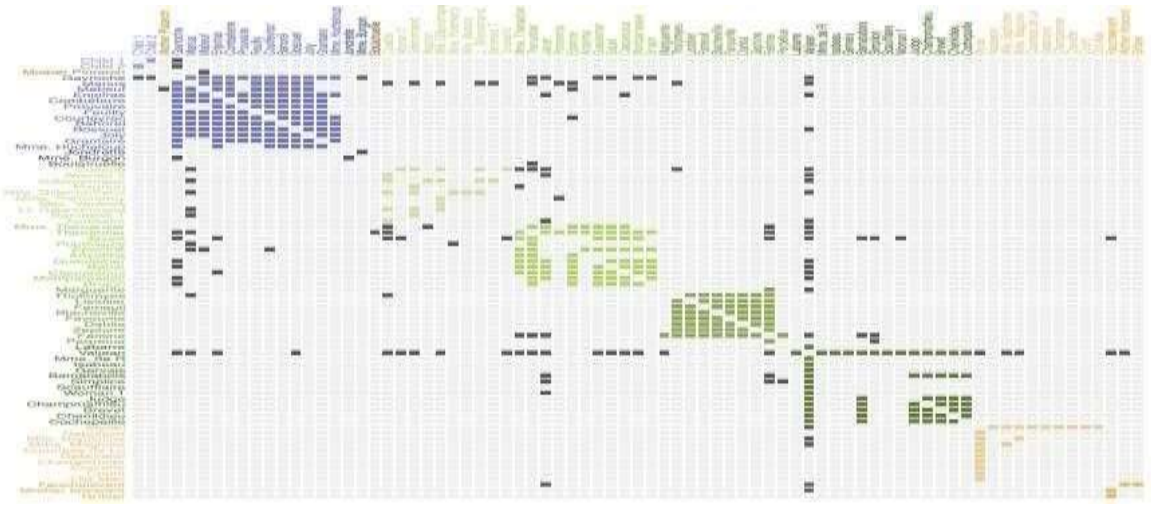
Examples:

general tree visualization

dendrogram



Network



Examples:

matrix

- **node-link diagram (link-based layout algorithm)**

Tableau:

Tableau is a Business Intelligence tool for visually analyzing the data. Users can create and distribute an interactive and shareable dashboard, which depict the trends, variations, and density of the data in the form of graphs and charts. Tableau can connect to files, relational and Big Data sources to acquire and process data. The software allows data blending and real-time collaboration, which makes it very unique. It is used by businesses, academic researchers, and many government organizations for visual data analysis. It is also

positionedas a leader Business Intelligence and Analytics Platform in Gartner Magic Quadrant.

Tableau Features:

Tableau provides solutions for all kinds of industries, departments, and data environments. Following are some unique features which enable Tableau to handle diverse scenarios.

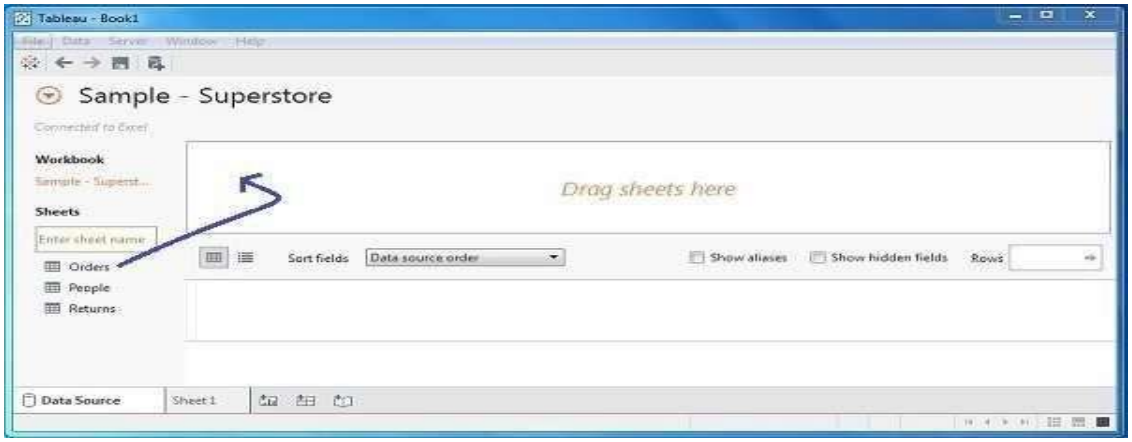
- **Speed of Analysis** − As it does not require high level of programming expertise, any user with access to data can start using it to derive value from the data.
- **Self-Reliant** − Tableau does not need a complex software setup. The desktop version which is used by most users is easily installed and contains all the features needed to start and complete data analysis.
- **Visual Discovery** − The user explores and analyzes the data by using visual tools likecolors, trend lines, charts, and graphs. There is very little script to be written as nearlyeverything is done by drag and drop.
- **Blend Diverse Data Sets** − Tableau allows you to blend different relational, semi structured and raw data sources in real time, without expensive up-front integration costs. The users don't need to know the details of how data is stored.
- **Architecture Agnostic** − Tableau works in all kinds of devices where data flows. Hence, the user need not worry about specific hardware or software requirements to use Tableau.
- **Real-Time Collaboration** − Tableau can filter, sort, and discuss data on the fly and embed a live dashboard in portals like SharePoint site or Salesforce. You can save your view of data and allow colleagues to subscribe to your interactive dashboards sothey see the very latest data just by refreshing their web browser.
- **Centralized Data** − Tableau server provides a centralized location to manage all of the organization's published data sources. You can delete, change permissions, add tags, and manage schedules in one convenient location. It's easy to schedule extract refreshes and manage them in the data server. Administrators can centrally define a schedule for extracts on the server for both incremental and full refreshes. There are three basic steps involved in creating any Tableau data analysis report. These three steps are-

- **Connect to a data source** − It involves locating the data and using an appropriatetype of connection to read the data.
- **Choose dimensions and measures** − This involves selecting the required columnsfrom the source data for analysis.
- **Apply visualization technique** − This involves applying required visualization methods, such as a specific chart or graph type to the data being analyzed.

For convenience, let's use the sample data set that comes with Tableau installation named sample – superstore.xls. Locate the installation folder of Tableau and go to **My Tableau Repository**. Under it, you will find the above file at **Datasources\9.2\en_US-US**.

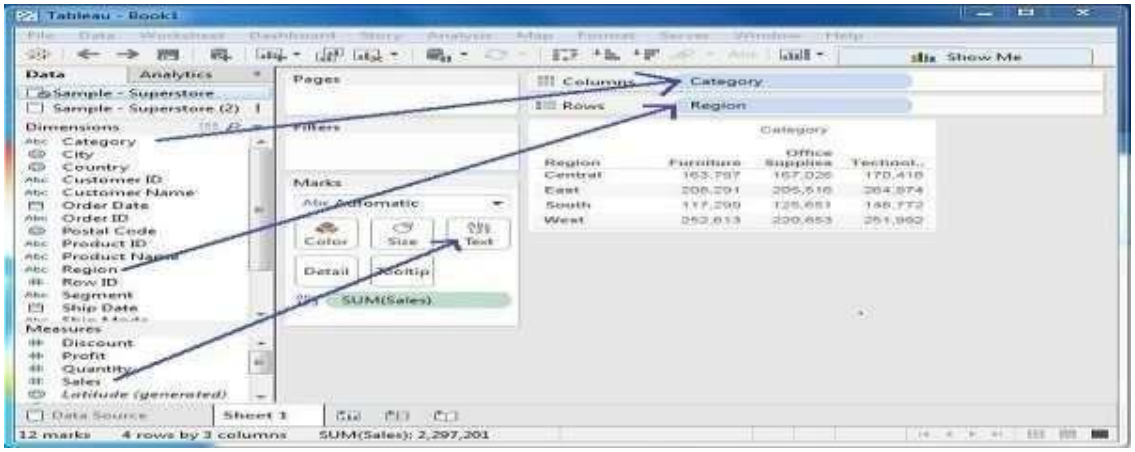
Connect to a Data Source

On opening Tableau, you will get the start page showing various data sources. Under the header **"Connect"**, you have options to choose a file or server or saved data source. Under Files, choose excel. Then navigate to the file **"Sample – Superstore.xls"** as mentioned above. The excel file has three sheets named Orders, People and Returns. Choose **Orders**.



Choose the Dimensions and Measures

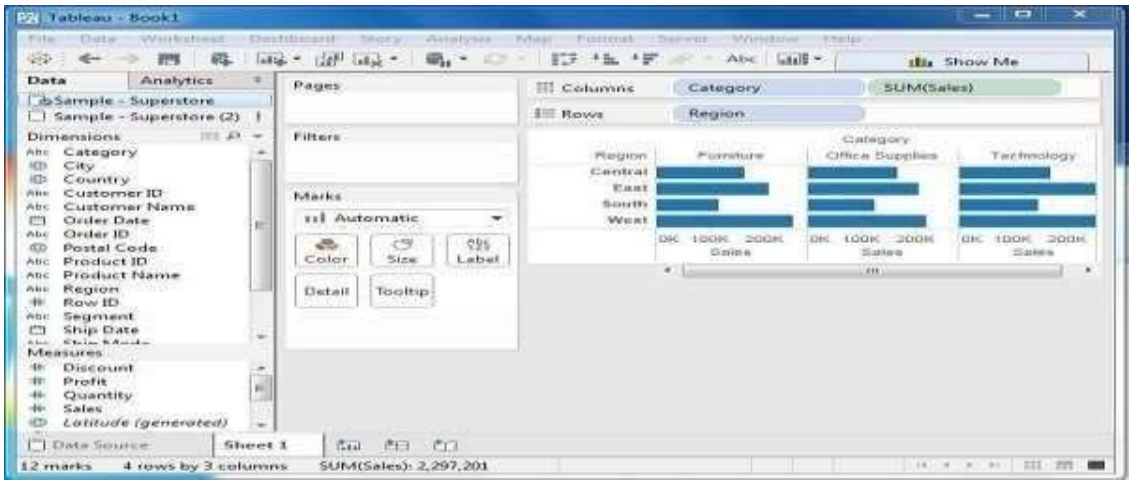
Next, choose the data to be analyzed by deciding on the dimensions and measures. Dimensions are the descriptive data while measures are numeric data. When put together, they help visualize the performance ofthe dimensional data with respect to the data which are measures. Choose **Category** and **Region** as the dimensions and **Sales** as the measure. Drag and drop them as shown in the following screenshot. The result shows the total sales in each category for each region



Apply Visualization Technique

In the previous step, you can see that the data is available only as numbers.

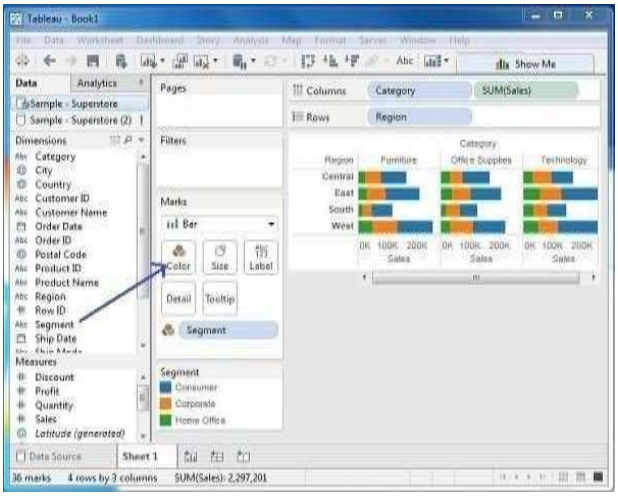
You have to readand calculate each of the values to judge the performance. However, you can see them as graphs or charts with different colors to make a quicker judgment.
We drag and drop the sum (sales) column from the Marks tab to the Columns shelf. The table showing the numeric values of sales now turns into a bar chart automatically.



You can apply a technique of adding another dimension to the existing data. This will add more colors to the existing bar chart as shown in the following screenshot.

Data visualization using Tableu:

1. **Iris Dataset**



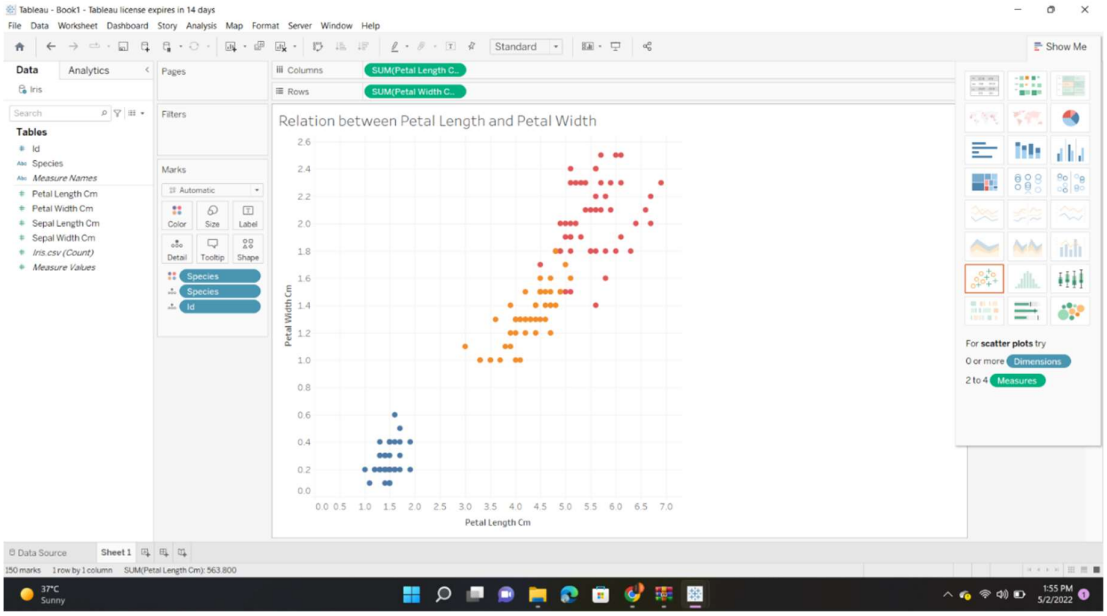


Tableau - Book1 - Tableau license expires in 14 days
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Show Me
Data
Analytics
Pages
Filters
Marks
Columns
SUM(Petal Length C...
Rows
SUM(Petal Width C...
Relation between Petal Length and Petal Width
Petal Width Cm
Petal Length Cm
Tables
Id
Species
Measure Names
Petal Length Cm
Petal Width Cm
Sepal Length Cm
Sepal Width Cm
Iris.csv (Count)
Measure Values
For scatter plots try
0 or more Dimensions
2 to 4 Measures
Data Source
Sheet 1
150 marks
1 row by 1 column
SUM(Petal Length Cm): 563.800
37°C
Sunny
1:55 PM
5/2/2022

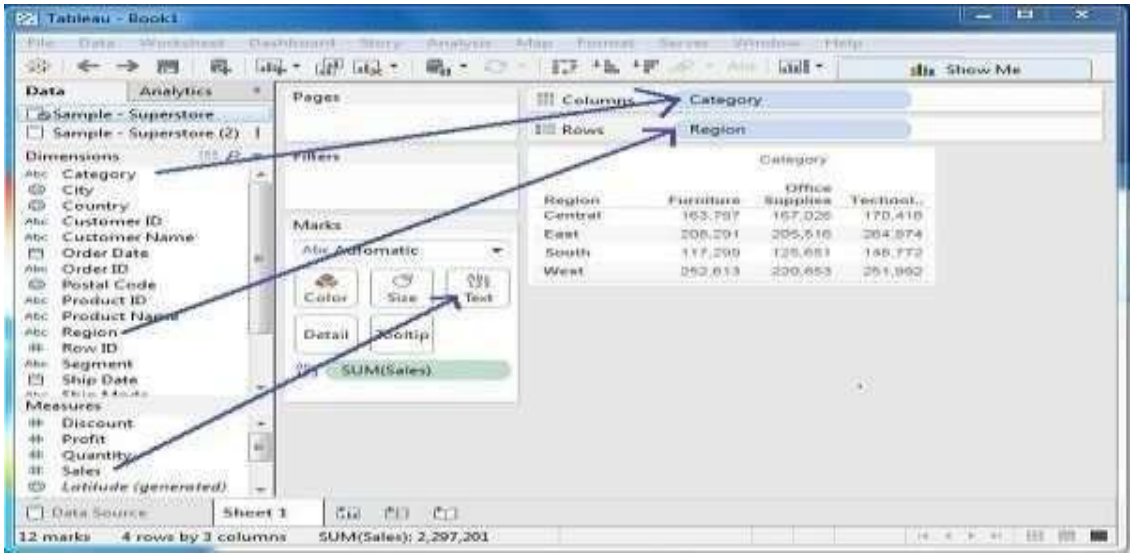


Tableau - Book1
Show Me
Data
Analytics
Sample - Superstore
Sample - Superstore (2)
Dimensions
Category
City
Country
Customer ID
Customer Name
Order Date
Order ID
Postal Code
Product ID
Product Name
Region
Row ID
Segment
Ship Date
Measures
Discount
Profit
Quantity
Sales
Latitude (generated)
Pages
Filters
Marks
Automatic
Color
Size
Text
Detail
Tooltip
SUM(Sales)
Columns
Category
Rows
Region
Category
Region
Furniture
Office Supplies
Technol..
Central 163,797 167,026 170,416
East 208,291 205,516 264,974
South 117,299 125,651 148,772
West 252,613 220,853 251,992
Data Source
Sheet 1
12 marks
4 rows by 3 columns
SUM(Sales): 2,297,201

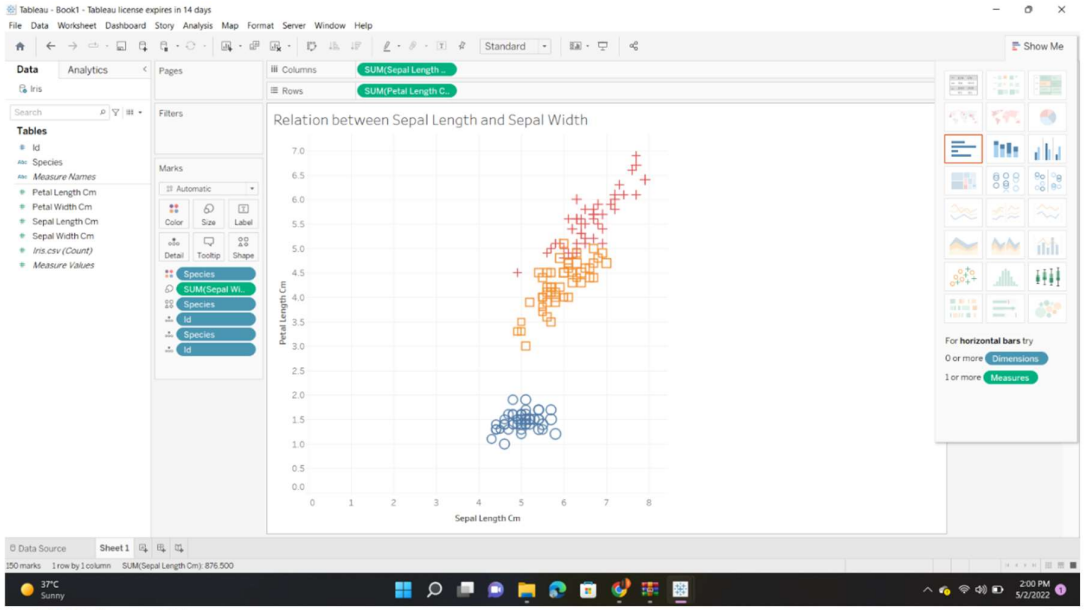


Tableau - Book1 - Tableau license expires in 14 days
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Standard
Show Me
Data
Analytics
Iris
Search
Tables
Id
Species
Measure Names
Petal Length Cm
Petal Width Cm
Sepal Length Cm
Sepal Width Cm
Iris.csv (Count)
Measure Values
Pages
Filters
Marks
Automatic
Color
Size
Label
Detail
Tooltip
Shape
Species
SUM(Sepal Wi..
Species
Id
Species
Id
Columns
SUM(Sepal Length ..
Rows
SUM(Petal Length C..
Relation between Sepal Length and Sepal Width
Petal Length Cm
Sepal Length Cm
For horizontal bars try
0 or more Dimensions
1 or more Measures
Data Source
Sheet 1
150 marks 1 row by 1 column SUM(Sepal Length Cm): 876.500
37°C
Sunny
2:00 PM
5/2/2022

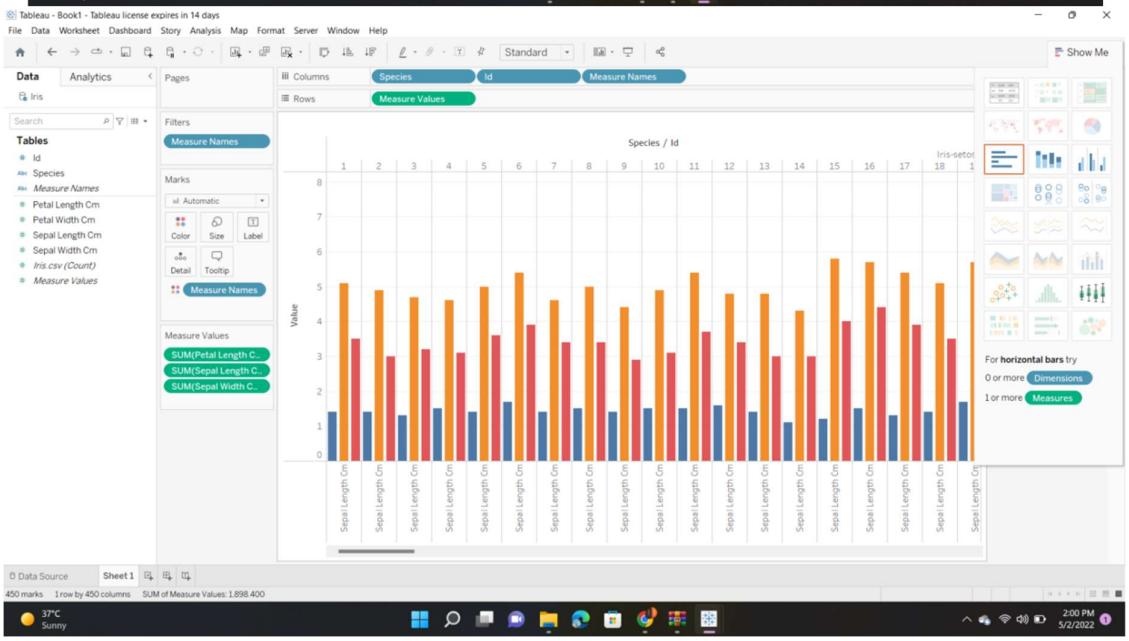
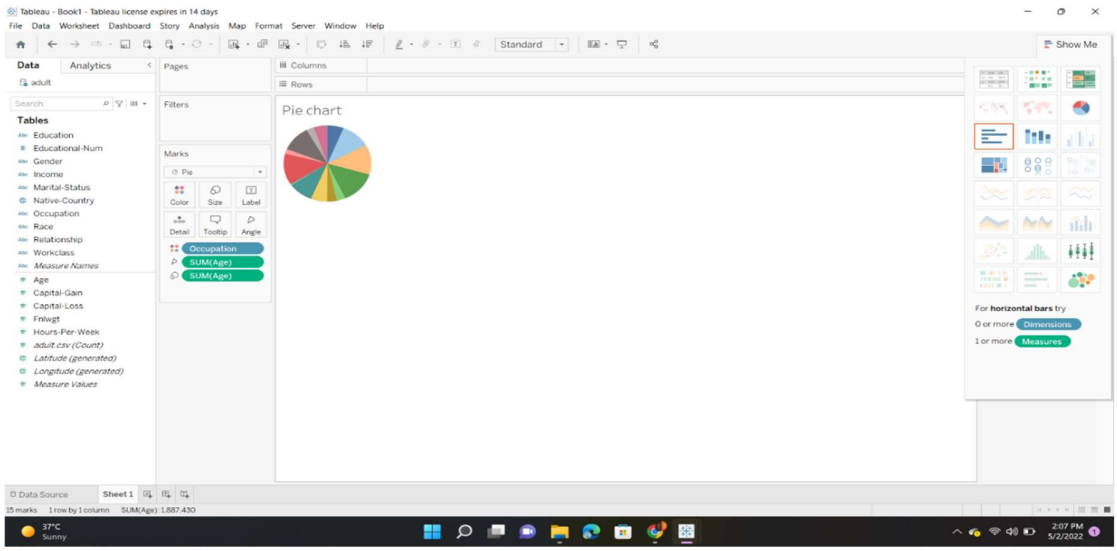
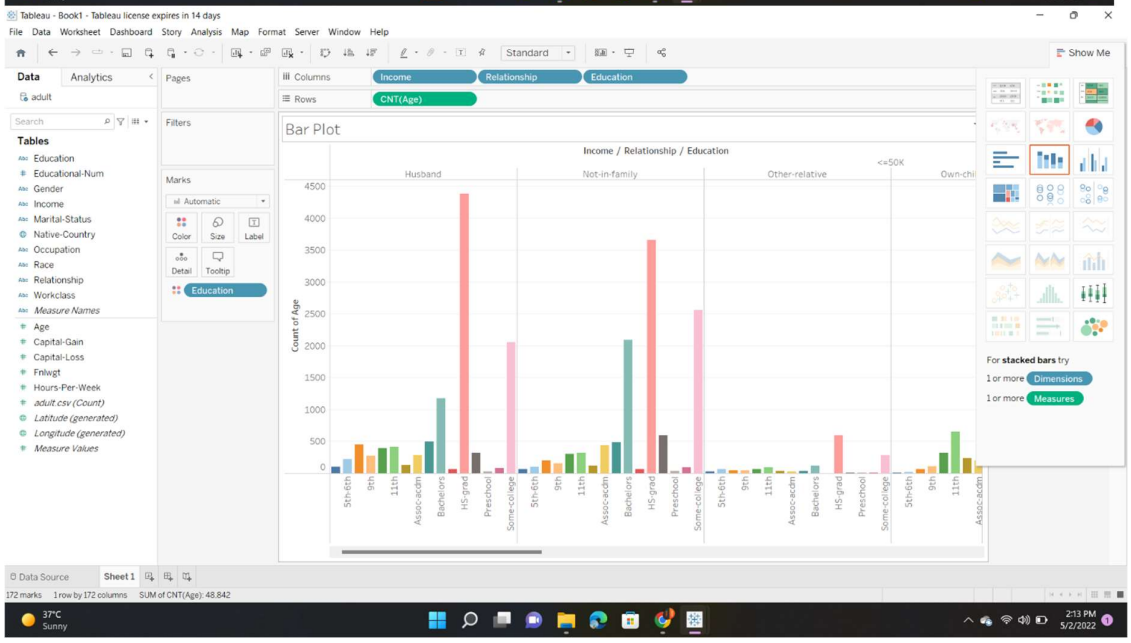
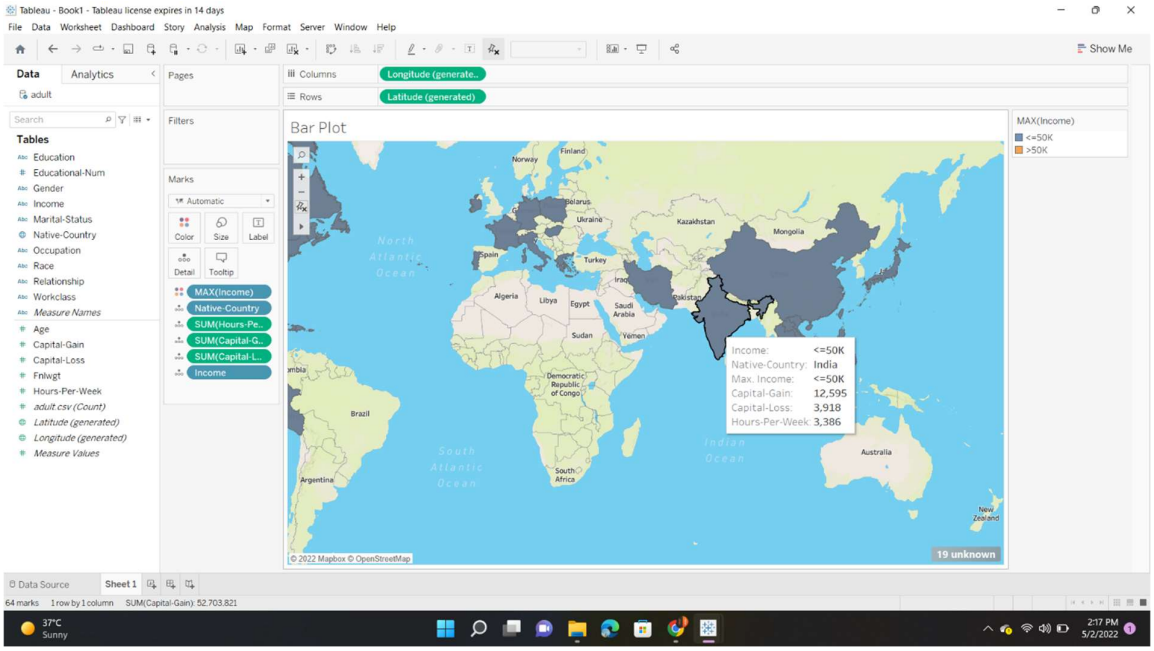


Tableau - Book1 - Tableau license expires in 14 days
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Standard
Show Me
Data
Analytics
Iris
Search
Tables
Id
Species
Measure Names
Petal Length Cm
Petal Width Cm
Sepal Length Cm
Sepal Width Cm
Iris.csv (Count)
Measure Values
Pages
Filters
Measure Names
Marks
Automatic
Color
Size
Label
Detail
Tooltip
Measure Names
Measure Values
SUM(Petal Length C..
SUM(Sepal Length C..
SUM(Sepal Width C..
Columns
Species
Id
Measure Names
Rows
Measure Values
Species / Id
Value
Sepal Length Cm
For horizontal bars try
0 or more Dimensions
1 or more Measures
Data Source
Sheet 1
450 marks 1 row by 450 columns SUM of Measure Values: 1,898.400
37°C
Sunny
2:00 PM
5/2/2022

Adult dataset :







Conclusion:

Thus we have learnt how to Visualize the data in different types (1 1D (Linear) Data visualization,2D (Planar) Data Visualization, 3D (Volumetric) Data Visualization, Temporal Data Visualization,Multidimensional Data Visualization, Tree/Hierarchical Data visualization, Network Data visualization) by using Tableau Software.