

**Visvesvaraya Technological University
Belagavi-590 018, Karnataka**



**A MINI PROJECT REPORT ON
“Indexing on Cricket Data Set”**

**Mini-Project Report submitted in Partial fulfillment of the requirement for the 6th
Semester File Structures Laboratory with Mini-Project
[17ISL68]**

**Bachelor of Engineering
In
Information Science and Engineering**

Submitted by

RAHUL VASISTA J V [1JT17IS032]

Under the Guidance of
Mr. Vadiraja A
Asst.Professor, Department of ISE



**Department of Information Science and Engineering
Jyothy Institute of Technology,
Tataguni, Bengaluru-56008**

Jyothy Institute of Technology,
Department of Information Science and Engineering
Tataguni, Bengaluru-560082



CERTIFICATE

Certified that the mini project work entitled “**Indexing on Cricket Data Set**” carried out by **RAHUL VASISTA J V [1JT17IS032]** bonafide student of Jyothy Institute of Technology, in partial fulfillment for the award of **Bachelor of Engineering in Information Science and Engineering** Department of the **Visvesvaraya Technological University, Belagavi** during the year **2020-2021**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Mr. Vadiraja A

Guide, Asst, Professor

Dept. Of ISE

Dr. Harshvardhan Tiwari

Associate Professor and HoD

Dept. OF ISE

External Viva Examiner

Signature with Date:

-
-

ACKNOWLEDGMENT

The satisfaction that accompanies the successful completion of my project Would be incomplete without mentioning the people who made it possible, I am thankful to **Dr. Gopalakrishna K**, Principal, JIT, Bangalore for Being kind enough to provide us an opportunity to work on a project in this Institution and I am thankful to **Dr. Harshvardhan Tiwari**, HoD, Dept of ISE for his co-operation and encouragement at all moments of our approach and Whose constant guidance and encouragement crowns all the efforts with Success. I would greatly mention enthusiastic influence provided by **Mr. Vadiraja A**, Asst. Professor, Dept. of ISE for his ideas and co-operation Showed on us during our venture and making this project a great success. Finally, it's pleasure and happiness to the friendly co-operation showed by all The staff members of Information Science Department, JIT.

RAHUL VASISTA J V
1JT17IS032

ABSTRACT

Indexing is a data structure technique to efficiently retrieve records from the database files based on some attributes on which the indexing has been done. Indexing in database systems is like what we see in books.

Primary Index is an ordered file which is fixed length size with two fields. The first field is the same as a primary key and second, field is pointed to that specific data block. In the primary Index, there is always one to one relationship between the entries in the index table.

The secondary Index can be generated by a field which has a unique value for each record, and it should be a candidate key. It is also known as a non-clustering index. This two-level database indexing technique is used to reduce the mapping size of the first level. For the first level, a large range of numbers is selected because of this; the mapping size always remains small.

I will be trying to work on 1 lakh records which is based on cricket county players of some countries. I will be building an index and then will perform some activities such as searching through index and modify and replacing the records adding an extra record to the file or maybe deleting some unwanted files in the Data Set.

This project has been created using PyCharm, with the platform WINDOWS. The project title- “Indexing” talks about how we can access csv file of almost One lakh record and perform several actions such as search modify add delete do the same actions also using the index.

TABLE OF CONTENTS

SERIAL NO.	DESCRIPTION	PAGE NO.
	Chapter 1	
1	Introduction	1 – 6
1.1	Introduction to File Structures	1
1.2	Introduction to File System	2
1.3	Indexing	3-6
	Chapter 2	
2	Requirement analysis and design	7-8
2.1	Domain Understanding	7
2.2	Classification of Requirements	7
2.3	System Analysis	7
	Chapter 3	
3	Implementation	9-14
	Chapter 4	
4	Result and Analysis	15 - 21

CHAPTER 1

INTRODUCTION

INTRODUCTION

1.1 Introduction to File Structures

In simple terms, a file is a collection of data stored on mass storage (e.g., disk or tape). But there is one important distinction that must be made at the outset when discussing file structures. And that is the difference between the logical and physical organization of the data.

Overall a file structure will specify the logical structure of the data, that is the relationships that will exist between data items independently of the way in which these relationships may be realized within any computer. It is this logical aspect that we will concentrate on. The physical organization is much more concerned with optimizing the use of the storage medium when a logical structure is stored on, or in it. Typically for every unit of physical store there will be several units of the logical structure (probably records) to be stored in it.

For example, if we were to store a tree structure on a magnetic disk, the physical organization would be concerned with the best way of packing the nodes of the tree on the disk given the access characteristics of the disk.

Like all subjects in computer science the terminology of file structures has evolved higgledy-piggledy without much concern for consistency, ambiguity, or whether it was possible to make the kind of distinctions that were important.

It was only much later that the need for a well-defined, unambiguous language to describe file structures became apparent. In particular, there arose a need to communicate ideas about file structures without getting bogged down by hardware considerations.

1.2 Introduction to File System

In computing, a file system or file system controls how data is stored and retrieved. Without a file system, information placed in a storage medium would be one large body of data with no way to tell where one piece of information stops and the next begins. By separating the data into pieces and giving each piece a name, the information is easily isolated and identified. Taking its name from the way paper-based information systems are named, each groups of data is called a “file”. The structure and logic rules used to manage the groups of information and their names is called a “file system”.

There are many kinds of file systems. Each one has different structure and logic, properties of speed, flexibility, security, size and more. Some file systems have been designed to be used for specific applications. For example, the ISO 9660 file system is designed specifically for optical discs.

File systems can be used on numerous different types of storage devices that use different kinds of media. The most common storage device in use today is a hard disk drive. Other kinds of media that are used include flash memory, magnetic tapes, and optical discs. In some cases, such as with tmpfs, the computer's main memory (random-access memory, RAM) is used to create a temporary file system for short-term use.

Some file systems are used on local data storage devices; others provide file access via a network protocol (for example, NFS, SMB, or 9P clients). Some file systems are “virtual”. Meaning that the supplied “files” (called virtual files) are computed on request (e.g. procfs) or are merely a mapping into a different file system used as a backing store. The file system manages access to both the content of files and the metadata about those files. It is responsible for arranging storage space; reliability, efficiency, and tuning regarding the physical storage medium are important design considerations.

1.3 Introduction to PYTHON

- Python is one of the most widely used programming language and its object-oriented approach is helpful in writing the clear and logical code for small scale as well as for the large-scale projects.
- It is considered to be a user-friendly programming language and it is known for its functional programming and the standard libraries.
- The main advantage of the language is that it uses white space indentation rather than commas or brackets or any kind of delimiters and it avoids usage of any type of semi colons at end of each line.

1.4 Introduction to Indexing

Indexing is a way to optimize the performance of a database by minimizing the number of disk accesses required when a query is processed. It is a data structure technique which is used to quickly locate and access the data in a database.

Indexes are created using a few database columns.

- The first column is the **Search key** that contains a copy of the primary key or candidate key of the table. These values are stored in sorted order so that the corresponding data can be accessed quickly.
Note: The data may or may not be stored in sorted order.
- The second column is the **Data Reference** or **Pointer** which contains a set of pointers holding the address of the disk block where that key value can be found.

1.4.1 Introduction to Simple Indexing:

- Simple Indexing is the process of accessing of lakhs of records using the primary index or the unique key of the record with the help of the starting location of the records.
- There is a separate file taken which is used to store the primary key i.e. the unique key and contains the location or the index of the same key.
- On performing the search query, the primary key corresponding to the searched key is looked in the index file and then by getting the location in the same file, the records file is accessed at that particular location and the line is displayed.
- Simple Indexing is also considered one of the efficient ways of accessing lakhs of records and is a faster way when dealing with time constraint.

1.4.2 Introduction to Secondary Indexing:

- A secondary index, put simply, is a way to efficiently access records in a database (the primary) by means of some piece of information other than the usual (primary) key.
- In Berkeley DB, this index is simply another database whose keys are these pieces of information (the secondary keys), and whose data are the primary keys. Secondary indexes can be created manually by the application; there is no disadvantage, other than complexity, to doing so.
- However, when the secondary key can be mechanically derived from the primary key and datum that it points to, as is frequently the case, Berkeley DB can automatically and transparently manage secondary indexes.

1.5 Introduction to B+ tree:

- Simple Indexing is the process of accessing of lakhs of records using the primary index or the unique key of the record with the help of the starting location of the records.
- There is a separate file taken which is used to store the primary key i.e. the unique key and contains the location or the index of the same key.
- On performing the search query, the primary key corresponding to the searched key is looked in the index file and then by getting the location in the same file, the records file is accessed at that particular location and the line is displayed.
- Simple Indexing is also considered one of the efficient way of accessing lakhs of records and is a faster way when dealing with time constraint.

1.6 Challenges faced during the Project

- The main challenge was after selecting the language i.e. Python. Even though Python is considered as a beginner's language and there was a need to deal with lakhs of records without using any of the built-in functions more than 24 hours was spent just to code 9 lines to build the index using primary key.
- B+ Tree is a built-in package in Python and hence there was no particular code defining the B+ Tree. The whole code including search function was implemented with the help of videos explaining the operation of B+ Tree.
- The building up of tree for few thousands of records is easy but when tree is to be built for lakhs of records, it involves lot of time but the searching for the elements takes 0.0 OR up to 1milliseconds.

1.7 Algorithm to build a Primary Index and B+ Tree:

Indexing is a way to optimize performance of a file system by minimizing the number of disk accesses required when a query is processed. An index is a data structure which is used to quickly locate and access the data in a disk of the file system.

Indexes are created using some columns in a file:

- The first column is the search key that contains a copy of the primary key or candidate key of the table. These values are stored in sorted order so that the corresponding data can be accessed quickly.
- The second column is the block where that particular key value can be found.

The data is sorted according to the search key. It includes sequential file organization. The primary key in data frame is used to create the index. As primary keys are unique and are stored in sorted manner, the performance of searching operation is quite efficient.

Algorithm:

```
int insert Sorted (int arr [], int n, int key, int capacity)
```

```
    If (n >= capacity)
```

```
        Return n;
```

```
    int i;
```

```
    For (i=n-1; (i >=0 && arr[i] >key); i--)
```

```
        arr [i+1] = arr[i];
```

```
    arr [i+1] = key;
```

```
    Return (n+1);
```

CHAPTER 2

REQUIREMENT ANALYSIS AND DESIGN

REQUIREMENT ANALYSIS AND DESIGN

2.1 Domain Understanding

The Main Objective of this Project is to perform different operations on the data set taken like search, modify, appending and deletion of the records.

2.2 Classification of Requirements

System Requirements

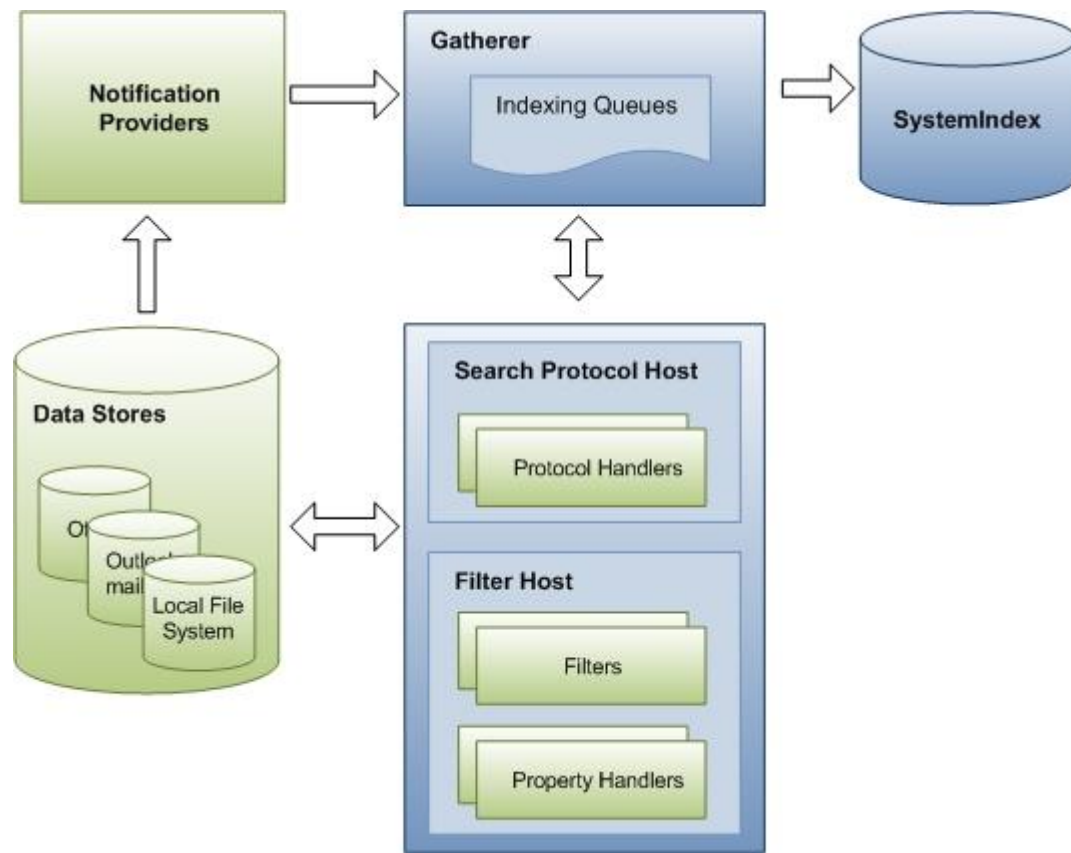
- OS: Windows
- Python Installed

Software and Hardware Requirements

Programming Languages: Python

2.1 System Analysis

When the program is executed, it asks the user whether he wants to search a record modify a record or does he have to add a record or delete a record. There is another option where the user can generate the index to another file and then search the record from the extra generated index. In the search option you can search the record by specifying the column name in which you want to search and proceed to search in any of the columns you need to access. In the modify option you can modify the records by specifying the index of the record and modify any column you want to modify and then continue with the other operations. In the appending option you can add records to the data set and in deletion you can delete unwanted records from the data set. We can also search records after copying the index file to another file and then by searching it from there.



CHAPTER 3

IMPLEMENTATION

IMPLEMENTATION

There are totally 6 main operations that can be performed on the data set. They are

3.1 SEARCH

```

• def searchById():
    start = time.time()
    ID = str(input('Enter the id you want to search\n'))
    csv_file = csv.reader(open('cricket_data.csv', 'r', encoding='utf-8', errors='ignore'))
    flag = 0
    for row in csv_file:
        if ID == row[0]:
            print(row)
            flag = 0
            break
        else:
            flag = 1
    if flag == 1:
        print('Not Found')
    end = time.time()
    print('Process Completed in', end - start, 'seconds')
def searchByName():
    start1 = time.time()
    NAME = (input('Enter the Name You Want Search'))
    csv_file = csv.reader(open('cricket_data.csv', 'r', encoding='utf-8', errors='ignore'))
    flag1 = 0
    for row in csv_file:
        if NAME == row[1]:
            print(row)
            flag1 = 0
            break
        else:
            flag1 = 1
    if flag1 == 1:
        print('Not Found')
    end1 = time.time()
    print('Process Completed in', end1 - start1, 'seconds')
def searchByCOUNTRY():
    start2 = time.time()
    COUNTRY = (input('Enter the Country You Want Search'))
    csv_file = csv.reader(open('cricket_data.csv', 'r', encoding='utf-8', errors='ignore'))
    flag = 0
    for row in csv_file:
        if COUNTRY == row[2]:
            print(row)

```

```

        flag = 0
        break
    else:
        flag = 1
if flag == 1:
    print('Not Found')
end2 = time.time()
print('Process Completed in', end2 - start2, 'seconds')
def searchByCurrentage():
    start3 = time.time()
    Currentage = (input('Enter the Age You Want Search'))
    csv_file = csv.reader(open('cricket_data.csv', 'r', encoding='utf-8', errors='ignore'))
    flag = 0
    for row in csv_file:
        if Currentage == row[4]:
            print(row)
            flag = 0
            break
        else:
            flag = 1
    if flag == 1:
        print('Not Found')
    end3 = time.time()
    print('Process Completed in', end3 - start3, 'seconds')
def searchByBattingstyle():
    start4 = time.time()
    Battingstyle = (input('Enter the Batting style You Want Search'))
    csv_file = csv.reader(open('cricket_data.csv', 'r', encoding='utf-8', errors='ignore'))
    flag = 0
    for row in csv_file:
        if Battingstyle == row[5]:
            print(row)
            flag = 0
            break
        else:
            flag = 1
    if flag == 1:
        print('Not Found')
    end4 = time.time()
    print('Process Completed in', end4 - start4, 'seconds')
def searchByBowlingstyle():
    start5 = time.time()
    Bowlingstyle = (input('Enter the Bowling style You Want Search'))
    csv_file = csv.reader(open('cricket_data.csv', 'r', encoding='utf-8', errors='ignore'))
    flag = 0
    for row in csv_file:
        if Bowlingstyle == row[6]:
            print(row)
            flag = 0
            break
        else:
            flag = 1
    if flag == 1:
        print('Not Found')
    end5 = time.time()
    print('Process Completed in', end5 - start5, 'seconds')
def search():
    print('Enter\n 1. Search by Id\n 2. Search by Name\n 3. Search by Country\n 4. Search by Age'
        '\n 5. Search by Batting Style \n 6. Search by Bowling Style\n')

```

```
src = int(input('Enter your choice\n'))
if src == 1:
    searchById()
elif src == 2:
    searchByName()
elif src == 3:
    searchByCOUNTRY()
elif src == 4:
    searchByCurrentage()
elif src == 5:
    searchByBattingstyle()
elif src == 6:
    searchByBowlingstyle()
else:
    print('Invalid Choice')
```

The Above program is written to search the dataset using different columns in the data set.

3.2 MODIFY

```

• def modify():
    start8 = time.time()
    df = pandas.read_csv('cricket_data.csv', index_col=False)
    Index = int(input("Enter your Index which is needed to be modified : "))
    print(df[df['ID'] == Index])
    Column = input("Enter your Column which is needed to be modified : ")
    print("OLD VALUE " + df.loc[df['ID'] == Index, Column])
    Replace = input("Enter your data to be Replaced : ")
    df.loc[df['ID'] == Index, Column] = Replace
    print("NEW VALUE " + df.loc[df['ID'] == Index, Column])
    df.to_csv('cricket_data.csv', index=False)
    end8 = time.time()
    print('Process Completed in', end8 - start8, 'seconds')

```

3.3 APPEND

```

• def append():
    start6 = time.time()
    print('All the values are necessary')
    with open('cricket_data.csv', 'a+', newline='') as f:
        fieldnames = ['ID', 'NAME', 'COUNTRY', 'Current age', 'Batting style', 'Bowling style']
        tgwrite = csv.DictWriter(f, fieldnames=fieldnames)
        idinput = int(input('Enter the Id'))
        nameinput = input('Enter the Name')
        countryinput = input('Enter th country')
        ageinput = int(input('Enter the age'))
        batinput = input('Enter the batting style')
        ballinput = input('Enter the bowling style')
        add = {'ID': idinput, 'NAME': nameinput, 'COUNTRY': countryinput, 'Current age': ageinput,
              'Batting style': batinput,
              'Bowling style': ballinput}
        tgwrite.writerow(add)
    end6 = time.time()
    print('Process Completed in', end6 - start6, 'seconds')

```

3.4 DELETION

```
def deletion():
    with open("cricket_data.csv", encoding='utf-8', errors='ignore') as csvfile:
        reader = csv.DictReader(csvfile)

        filePath = "cricket_data.csv"
        with open(filePath, 'w', newline='') as csvfile:
            fieldnames = ['ID', 'NAME', 'COUNTRY', 'Current age', 'Batting style', 'Bowling style']
            writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

            print("Delete an avatar")
            deletion1 = input("Enter avatar id to delete: ")

        for row in reader:
            if deletion1 != row['ID']:
                writer.writerow(row); # write all non-matching rows
            else:
                print("Avatar Record Deleted")
```

3.5 GENERATING AN INDEX FILE

```
def indexgen():
    with open("cricket_data.csv", "r", encoding='utf-8', errors='ignore') as pack:
        with open("cricket_data1.csv", "w", encoding='utf-8', errors='ignore') as index:
            count = 0
            for line in pack:
                stri = line[:8]
                index.write(stri)
                c = str(count)
                index.write(c + "\n")
                count = len(line) + cou
```

3.6 SEARCHING USING INDEX FILE

```
def searchByIndex():
    with open("cricket_data.csv", "r", encoding='utf-8', errors='ignore') as main1:
        with open("cricket_data1.csv", "r", encoding='utf-8', errors='ignore') as index1:
            lines = main1.readlines()
            search1 = input("Enter the Id")
            start2 = time.time()
            for line1 in index1:
                if search1 in line1:
                    print("The line is here in index file:" + line1)
                    line1 = line1.split(",")
                    loc = line1[1].split()
                    print(loc)
                    locate = int(loc[1])
                    x = main1.seek(locate)
                    y = main1.readlines(locate)
                    result = [ind for ind in y if search1 in ind]
                    result1 = result[0].split(',')
                    print(result1)
                    break
```

3.7 MENU

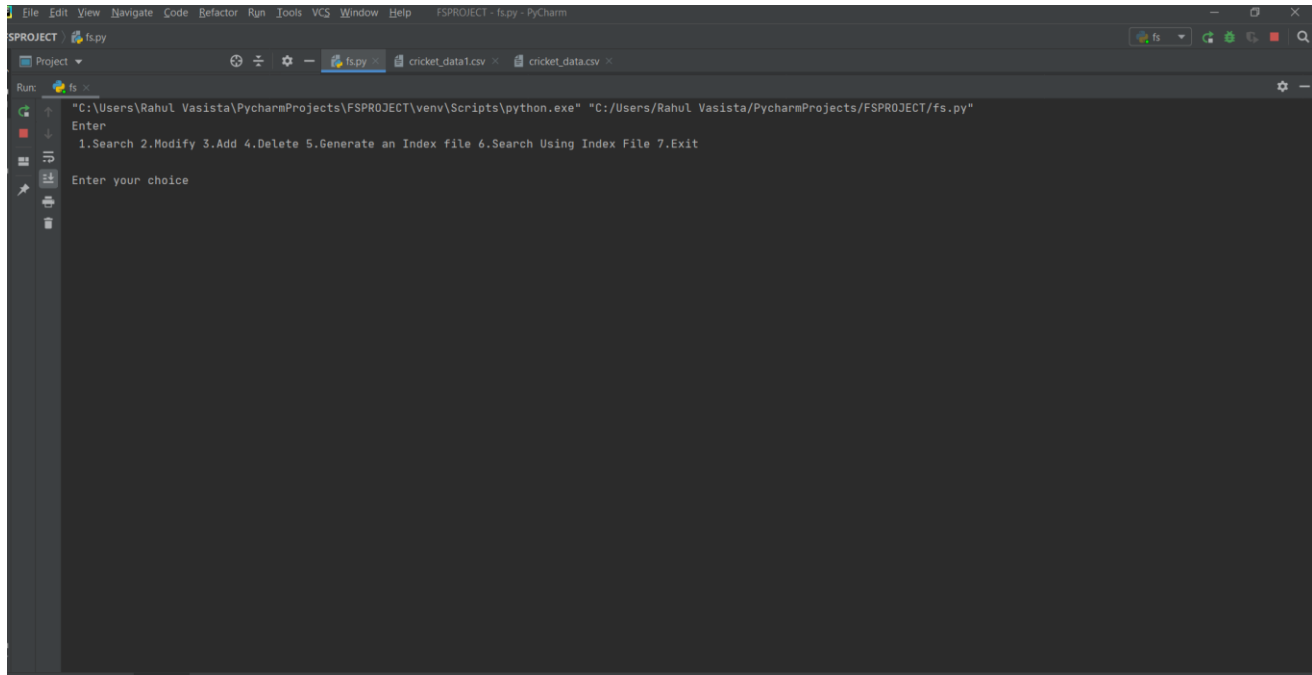
```
def main():
    while True:
        print('Enter \n 1.Search 2.Modify 3.Add 4.Delete 5.Generate an Index file 6.Search Using Index File 7.Exit\n')
        Choice = int(input('Enter your choice\n'))
        if Choice == 1:
            search()
        elif Choice == 2:
            modify()
        elif Choice == 3:
            append()
        elif Choice == 4:
            deletion()
        elif Choice == 5:
            indexgen()
        elif Choice == 6:
            searchByIndex()
        elif Choice == 7:
            print("Thanks")
            exit(0)
        else:
            print('Invalid Choice')
```

CHAPTER 4

RESULT AND ANALYSIS

RESULT AND ANALYSIS

4.1 MENU:



MENU

The Above Figure Is the Menu where the user can choose which operation, he should perform on the data set.

4.2 SEARCH MENU:

```

C:\Users\Rahul Vasista\PycharmProjects\FSPROJECT\venv\Scripts\python.exe "C:/Users/Rahul Vasista/PycharmProjects/FSPROJECT/fs.py"
Enter
1.Search 2.Modify 3.Add 4.Delete 5.Generate an Index file 6.Search Using Index File 7.Exit
Enter your choice
6
Enter
1. Search by Id
2. Search by Name
3. Search by Country
4. Search by Age
5. Search by Batting Style
6. Search by Bowling Style
Enter your choice
1

```

When you try to search the menu asks you to choose the column where you want to search and gives the output accordingly.

4.3 SEARCH RESULTS:

```

C:\Users\Rahul Vasista\PycharmProjects\FSPROJECT\venv\Scripts\python.exe "C:/Users/Rahul Vasista/PycharmProjects/FSPROJECT/fs.py"
Enter
1.Search 2.Modify 3.Add 4.Delete 5.Generate an Index file 6.Search Using Index File 7.Exit
Enter your choice
6
Enter
1. Search by Id
2. Search by Name
3. Search by Country
4. Search by Age
5. Search by Batting Style
6. Search by Bowling Style
Enter your choice
1
Enter the id you want to search
500
['500', 'Zeeshan Jamil', 'Pakistan', 'Zeeshan Jamil', '31 years 140 days', 'Right-hand bat', 'Right-arm offbreak']
Process Completed in 3.268505811691284 seconds
Enter
1.Search 2.Modify 3.Add 4.Delete 5.Generate an Index file 6.Search Using Index File 7.Exit
Enter your choice
1

```

Search menu gives result based on the user input and displays the record in the type of a list with the time it used to perform the specific operation

4.4 MODIFY:

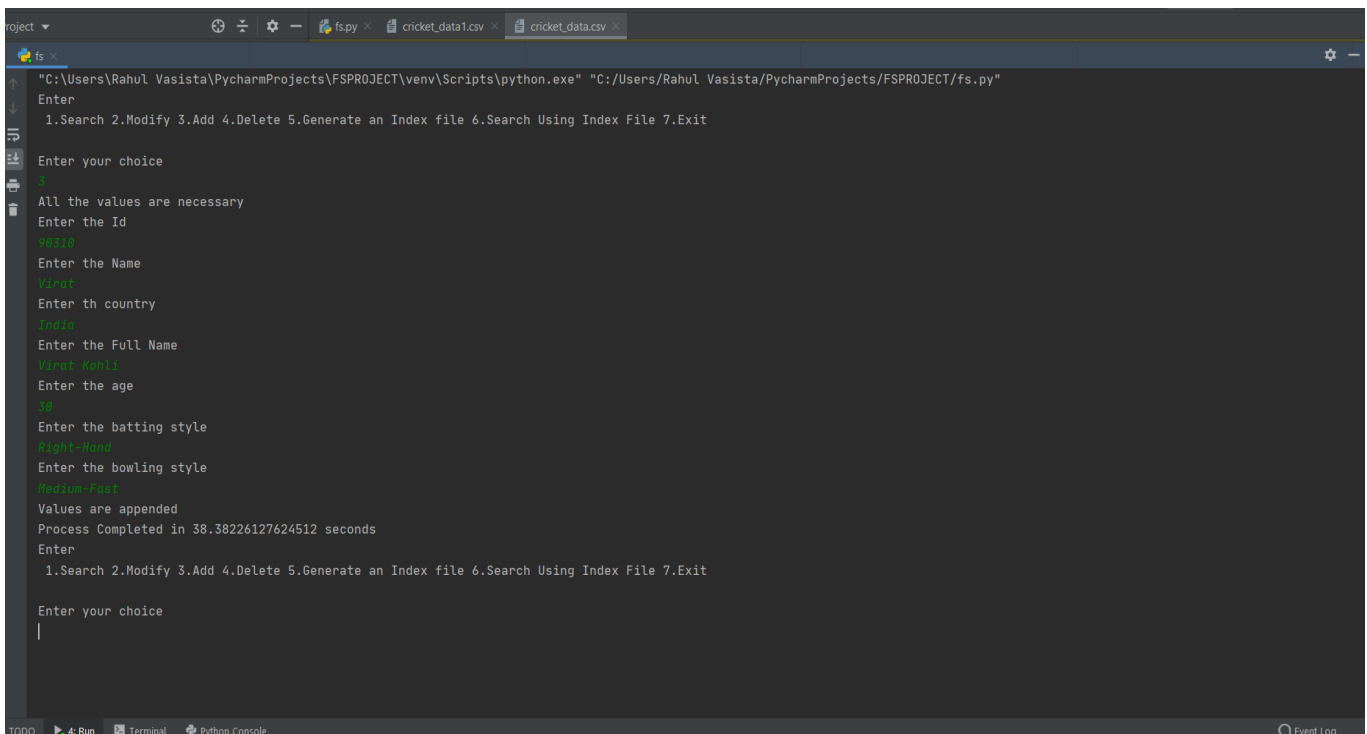
```
"C:\Users\Rahul Vasista\PycharmProjects\FSPROJECT\venv\Scripts\python.exe" "C:/Users/Rahul Vasista/PycharmProjects/FSPROJECT/fs.py"
Enter
1.Search 2.Modify 3.Add 4.Delete 5.Generate an Index file 6.Search Using Index File 7.Exit

Enter your choice
2
Enter your Index which is needed to be modified : 90309
ID  NAME COUNTRY Full name Current age Batting style Bowling style
90309 90310 Virat India 30 Virat Kohli Right-Hand Medium-Fast
Enter your Column which is needed to be modified : COUNTRY
90309 OLD VALUE India
Name: COUNTRY, dtype: object
Enter your data to be Replaced : ROYAL CHALLENGERS BANGLORE
90309 NEW VALUE ROYAL CHALLENGERS BANGLORE
Name: COUNTRY, dtype: object
Process Completed in 32.627681016922 seconds
Enter
1.Search 2.Modify 3.Add 4.Delete 5.Generate an Index file 6.Search Using Index File 7.Exit

Enter your choice
|
```

In modify option you can modify the old records using the index and modify the records as you require.

4.5 APPEND:



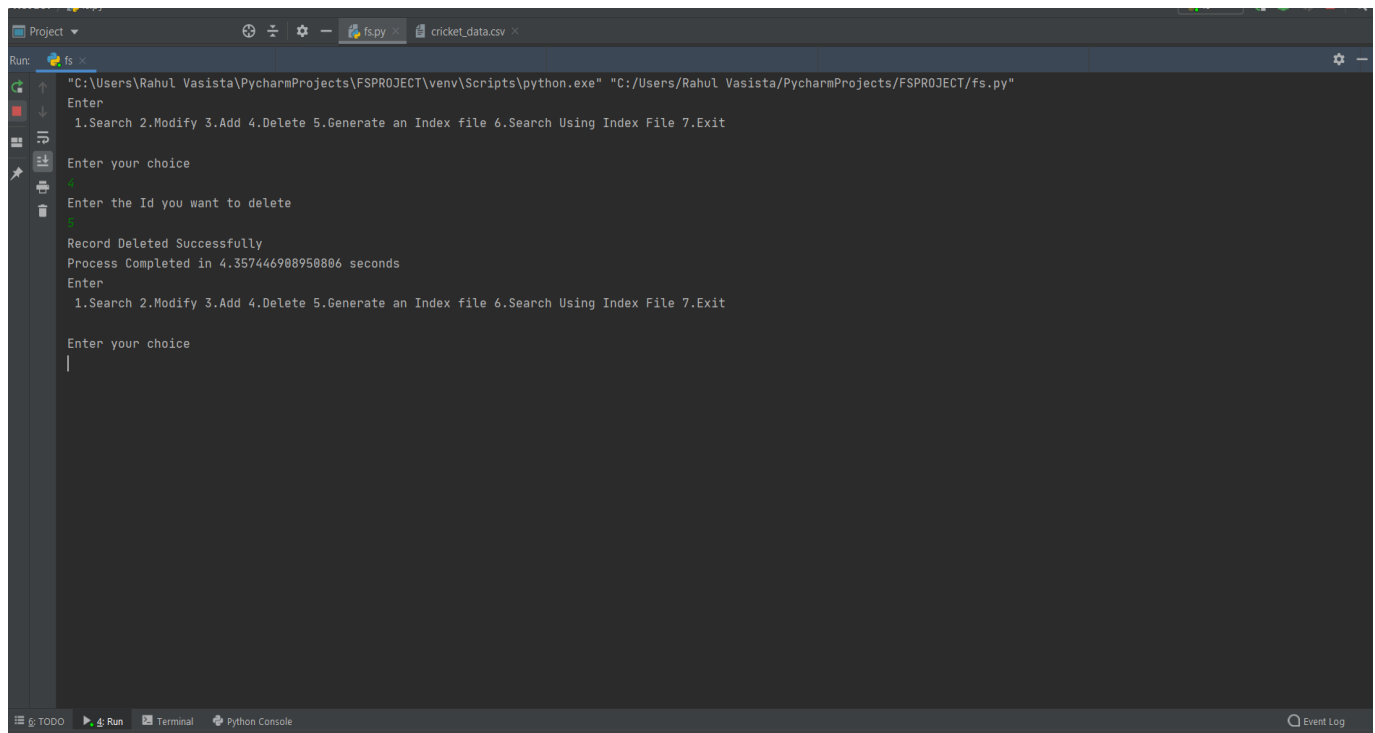
```
project
fs.py x cricket_data1.csv x cricket_data.csv x
fs.py x
"C:\Users\Rahul Vasista\PycharmProjects\FSPROJECT\venv\Scripts\python.exe" "C:/Users/Rahul Vasista/PycharmProjects/FSPROJECT/fs.py"
Enter
1.Search 2.Modify 3.Add 4.Delete 5.Generate an Index file 6.Search Using Index File 7.Exit

Enter your choice
3
All the values are necessary
Enter the Id
90310
Enter the Name
Virat
Enter th country
India
Enter the Full Name
Virat Kohli
Enter the age
30
Enter the batting style
Right-Hand
Enter the bowling style
Medium-Fast
Values are appended
Process Completed in 38.38226127624512 seconds
Enter
1.Search 2.Modify 3.Add 4.Delete 5.Generate an Index file 6.Search Using Index File 7.Exit

Enter your choice
|
```

In the above two Figure you can add the records to the data set.

4.6 DELETION:



```
"C:\Users\Rahul Vasista\PycharmProjects\FSPROJECT\venv\Scripts\python.exe" "C:/Users/Rahul Vasista/PycharmProjects/FSPROJECT/fs.py"
Enter
1.Search 2.Modify 3.Add 4.Delete 5.Generate an Index file 6.Search Using Index File 7.Exit

Enter your choice
4

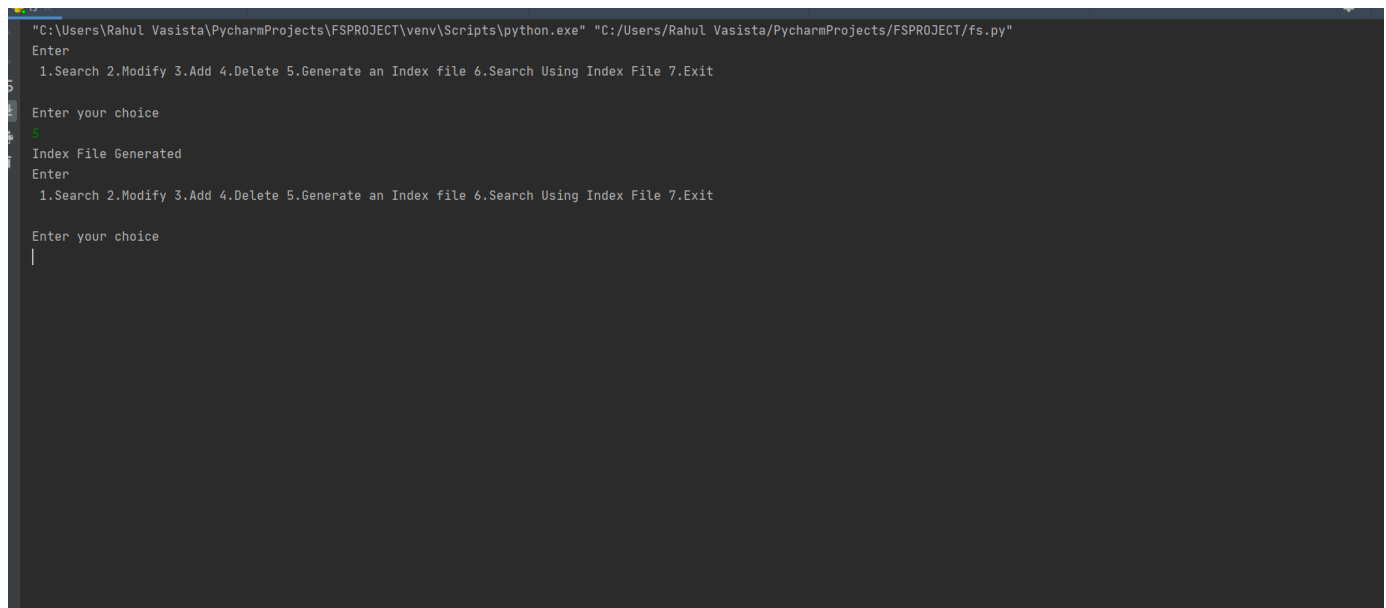
Enter the Id you want to delete
10

Record Deleted Successfully
Process Completed in 4.357446988958886 seconds
Enter
1.Search 2.Modify 3.Add 4.Delete 5.Generate an Index file 6.Search Using Index File 7.Exit

Enter your choice
|
```

In this option you can delete the unwanted records.

4.7 GENERATING AN INDEX FILE:



```
"C:\Users\Rahul Vasista\PycharmProjects\FSPROJECT\venv\Scripts\python.exe" "C:/Users/Rahul Vasista/PycharmProjects/FSPROJECT/fs.py"
Enter
1.Search 2.Modify 3.Add 4.Delete 5.Generate an Index file 6.Search Using Index File 7.Exit

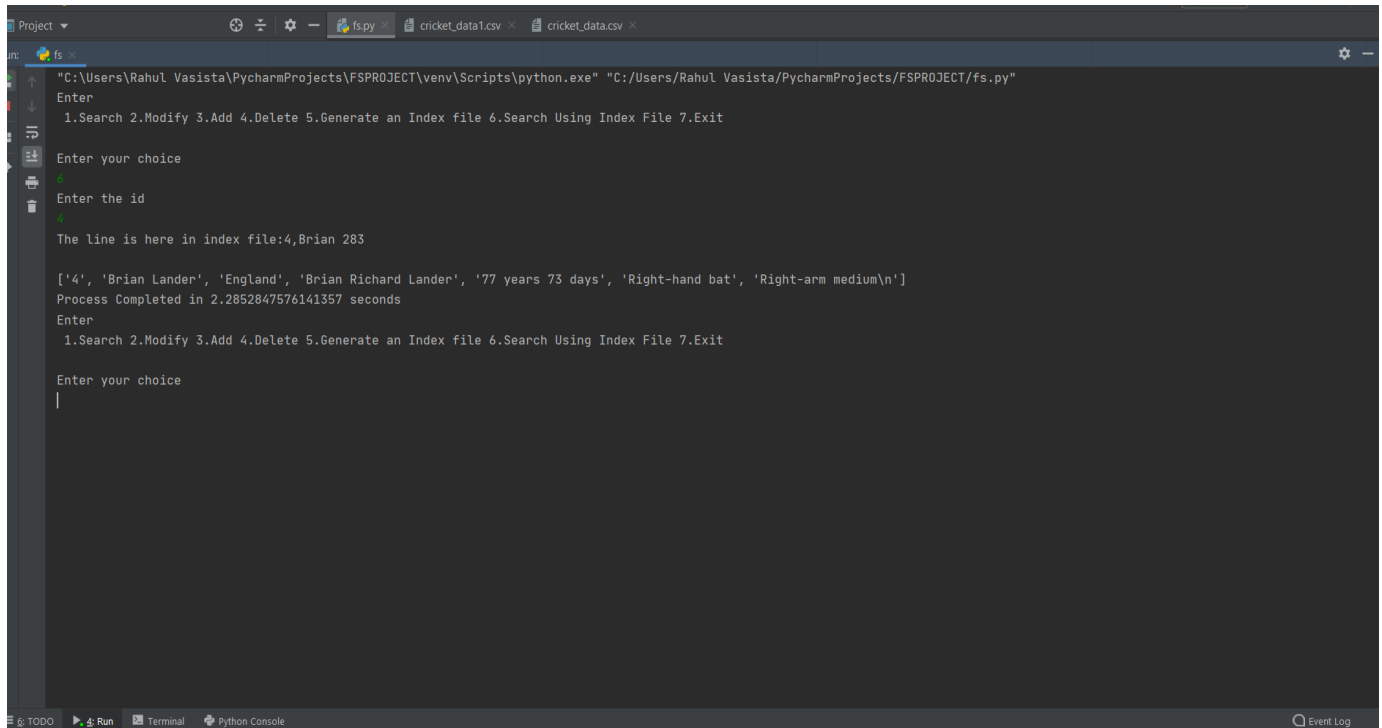
Enter your choice
5

Index File Generated
Enter
1.Search 2.Modify 3.Add 4.Delete 5.Generate an Index file 6.Search Using Index File 7.Exit

Enter your choice
|
```

Index file will be generated when you select the 5th option it uses a combination and generate an index and store in a separate file.

4.8 SEARCH USING AN INDEX FILE:



```

"C:\Users\Rahul Vasista\PycharmProjects\FSPROJECT\venv\Scripts\python.exe" "C:/Users/Rahul Vasista/PycharmProjects/FSPROJECT/fs.py"
Enter
1.Search 2.Modify 3.Add 4.Delete 5.Generate an Index file 6.Search Using Index File 7.Exit

Enter your choice
6
Enter the id
4
The line is here in index file:4,Brian 283

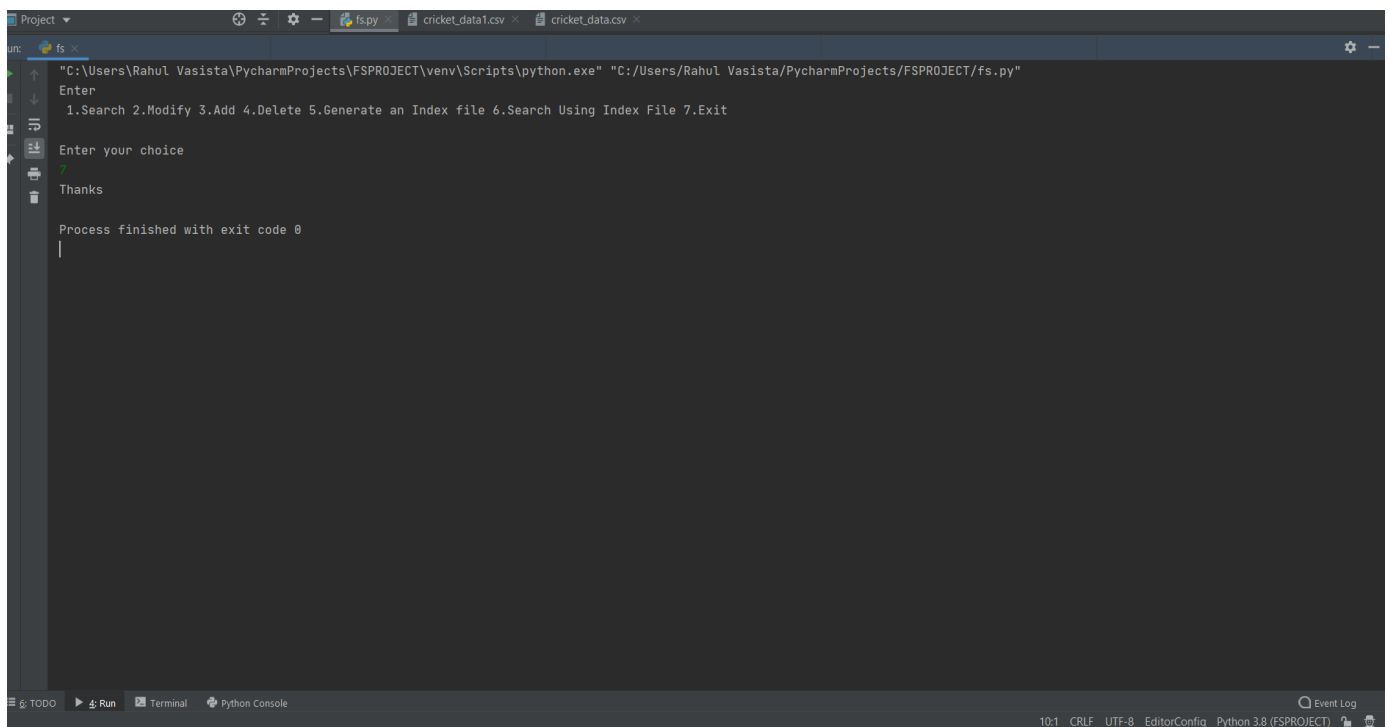
['4', 'Brian Lander', 'England', 'Brian Richard Lander', '77 years 73 days', 'Right-hand bat', 'Right-arm medium\n']
Process Completed in 2.2852847576141357 seconds
Enter
1.Search 2.Modify 3.Add 4.Delete 5.Generate an Index file 6.Search Using Index File 7.Exit

Enter your choice
|

```

The Search searches using the index created in the other file using that it accesses the file and searches accordingly.

4.9 EXIT:



```

"C:\Users\Rahul Vasista\PycharmProjects\FSPROJECT\venv\Scripts\python.exe" "C:/Users/Rahul Vasista/PycharmProjects/FSPROJECT/fs.py"
Enter
1.Search 2.Modify 3.Add 4.Delete 5.Generate an Index file 6.Search Using Index File 7.Exit

Enter your choice
7
Thanks

Process finished with exit code 0
|

```

4.10 Graph (Time vs Record Size)



4.11 CONCLUSION:

We have successfully used various functionalities of Python and were able to deal with lakhs of records.

- Indexes form an important part of designing, creating and testing information.
- Users search in a hurry for information to help them and give up after two or three tries.
- An index can point the way in harmony with user expectations or not.
- Indexing is an interactive analysis and creative process throughout the entire documentation.
- Implementation of B+ Tree is one of the efficient and faster way of searching the records compared to search using indexing.

Features:

1. Clean separation of various components to facilitate easy modification and revision.
2. All the data is maintained in a separate file to facilitate easy modification.
3. All the data required for different operations is kept in a separate file.

4.12 REFERENCES:

- www.data.gov.in

The dataset was downloaded from the above URL which contained various datasets from every domain.

- www.youtube.com – tutorialspoint

The channel helped in understanding how the B+ Tree operations work i.e. insertion and searching which helped in building the code for B+ Tree.

- www.stackoverflow.com

One of the best websites when stuck in building the codes using Python and helped lot when faced with exceptions

- www.javatpoint.com

This website contains all the information that is needed for understanding the working of B+ Tree and the operations that happens on the same.

- www.geeksforgeeks.com

This also helped a lot in understanding the basic concepts of the B+ Tree

- <https://www.cs.usfca.edu/~galles/visualization/BPlusTree.html>

Probably one of the best sites which gave a visualization about how the B+ Tree operations work.

- Michael J. Folk, Bill Zoellick, Greg Riccardi: File Structures-An Object Oriented Approach with C++, 3rd Edition, Pearson Education, 1998.
This book helped in understanding the construction of B+ Tree in an Object-Oriented Language

- <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4800951/>
Journals on Indexing

- https://www.foo.be/docs/tpj/issues/vol2_4/tpj0204-0010.html
Journals on B-tree