

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



A Mini Project Report on

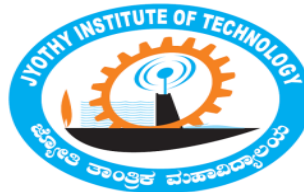
“ INDEXING ”

**MiniProject Report submitted in partial fulfilment of the
requirement for the
FILE STRUCTURE LAB[17ISL68]**

**Bachelor of Engineering
in
Information Science and Engineering**

**Submitted by
NEHA M S [1JT17IS024]**

**Under the guidance of
Mr. Vadiraja A
Assistant prof Department of ISE**



**Department of Information Science and Engineering
Jyothy Institute of Technology
Tataguni, Bengaluru-560082
2019-2020**

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



CERTIFICATE

Certified that the mini project work entitled “ **Indexing** ” carried out by **NEHA M S [1JT17IS024]** bonafide student of Jyothy Institute of Technology, in partial fulfilment for the award of **Bachelor of Engineering in Information Science and Engineering** department, under **Visvesvaraya Technological University, Belagavi** during the year **2019-2020**. 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
 Professor and HOD
 Dept. Of ISE

External Viva Examiner

Signature with Date :

- 1.
- 2.

ACKNOWLEDGEMENT

Firstly, we are very grateful to this esteemed institution “**Jyothy Institute of Technology**” for providing us an opportunity to complete our project.

We express our sincere thanks to our Principal **Dr.Gopalakrishna K** for providing us with adequate facilities to undertake this project.

We would like to thank **Dr. Harshvardhan Tiwari Professor and Head** of information Science and Engineering Department for providing for his valuable support.

We would like to thank our guide **Mr.Vadiraja A, Assistant .prof** for his keen interest and guidance in preparing this project work.

Finally, we would thank all our friends who have helped us directly or indirectly in this project.

NEHA M S [1JT17IS024]

ABSTRACT

File Structures is the Organization of Data in Secondary Storage Device in such a way it minimizes the time required to access and the space required to store data. A File Structure allows applications to read, write and modify data. Indexing is a way to enhance and optimize the performance by reducing the number the disk accesses. Indexing is a data structure technique to efficiently retrieve data based on some attributes on which the indexing has been done. The data file is ordered on a key field. The key field is generally the primary key of the relation. The dataset selected for this project can be categorized under "Sales" category. The datasets are generated through random logic in VBA(Visual Basic Application) and is downloaded from EforExcel. These are not real sales data and used only for the purpose of this project. The data file contains Order ID, Unit Price, Item Type, Region, Country, and Sales Channel. The main purpose of this project is to build an index for the given dataset. The Primary Index ranging from 0 to 99999 (1 Lakh Records) and Secondary Index - Region. We can Search using either primary index or secondary index, insert a new record, delete an existing record efficiently.

Table of Contents

Sl No.	Description	Page
1	Introduction	7
1.1	Introduction to File Structure	7
1.2	Introduction to Python	8
1.3	Introduction to Indexing	8
1.4	Scope and Importance of Work	9
2	Design	10
2.1	Primary Index	11
2.2	Secondary Index	11
2.3	Inserting	12
2.4	Searching	12
2.5	Time Complexity	13
3	Implementation	14
3.1	Algorithm to Build an Index	15
3.2	Algorithm for Insertion	15
3.3	Algorithm for Searching	15
3.4	Space Complexity	16
4	Results and Snapshots	19
	Conclusion	25
	References	25

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

1.1 Introduction to File Structure

A disk's relatively slow access time and the enormous, non-volatile capacity is the driving force behind file structure design. File structure should give access to all the capacity without making the application spend a lot of time waiting for the disk. File structure is a combination of representation for data in files and of operations for accessing the data.

A File Structure allows applications to read, write and modify data. It also supports finding the data that matches some search criteria or reading through the data in some particular order.

Efficiency of FS design for a particular application is decided on,

- Details of the representation of the data
- Implementation of the operations

A large variety in the types of data and in the needs of application makes FS design important. What is best for one situation may be terrible for other.

A file system is a process that manages how and where data on a storage disk, typically a hard disk drive (HDD), is stored, accessed and managed. It is a logical disk component that manages a disk's internal operations as it relates to a computer and is abstract to a human user. Regardless of type and usage, a disk contains a file system and information about where disk data is stored and how it may be accessed by a user or application. A file system typically manages operations, such as storage management, file naming, directories/folders, metadata, access rules and privileges.

1.2 Introduction to Python

- Python is an interpreted high-level programming language for general-purpose programming. This was created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.
- Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object oriented, imperative, functional and procedural, and has a large and comprehensive standard library.
- Python interpreters are available for many operating systems. C Python, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. C Python is managed by the non-profit Python Software Foundation.

1.3 Introduction to Indexing

A structure containing a set of entries each consisting of a key field and reference field, which is used to locate records in a data file. Key field is the part of an index which contains keys. Reference field is the part of an index which contains information to locate records. Simple indexing can be useful when the entire index can be held in memory. Changes (addition and deletions) require both the index and the data to be changed. Updates affect the index if the key field is changed, or if the record is moved. An update which moves a record can be handled as a deletion followed by addition.

Indexing can be of two types Primary index and Secondary index. Primary index is defined on an ordered data file. The data file is ordered on a key field. The key field is generally the primary key of the relation. Secondary index can be generated from a field which is the candidate key and has a unique value in every record or a non key with a duplicate values.

1.4 Scope and Importance of Work

- Data analysis is a process used to inspect, clean, transform and remodel data with a view to reach to a certain conclusion for a given situation. Data analysis helps in structuring the findings from different sources of data.
- Data analysis is very helpful in breaking a macro problem into micro parts.
- Data analysis acts like a filter when it comes to acquiring meaningful insights out of huge data set.
- Data analysis helps in keeping human bias away from the research conclusion with the help of proper statistical treatment.

When discussing data analysis it is important to mention that a methodology to analyse data needs to be picked. if a specific methodology is not selected data can neither be collected nor analyzed. The methodology should be present in the dissertation as it enables the reader to understand which methods have been used during the research and what type of data has been collected and analyzed throughout the process. The dissertation also presents a critical analysis of various methods and techniques that were considered but ultimately not used for the data analysis. An effective research methodology leads to better data collection and analysis and leads the researcher to arrive at valid and logical conclusions in the research. Without a specific methodology, observations and findings in a research cannot be made which means methodology is an essential part of a research or dissertation.

User will be able analysis time taken to build the index. This in turn will help them in design the index based on the time taken for the index to build.. Data analysis plays a very major in any of the project we do.

CHAPTER 2

DESIGN

2.1 Primary Index:

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 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 (Note that the data may or may not be stored in sorted order).
- The second column is the Data Reference which contains a set of pointers holding the address of the disk block where that particular key value can be found.

Clustering index is defined on an ordered data file. The data file is ordered on a non-key field. In some cases, the index is created on non-primary key columns which may not be unique for each record. In such cases, in order to identify the records faster, we will group two or more columns together to get the unique values and create index out of them. This method is known as clustering index. Basically, records with similar characteristics are grouped together and indexes are created for these groups.

The data is sorted according to the search key. It induces sequential file organisation. 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.

2.2 Secondary Index:

Secondary index may be generated from a field which is a candidate key and has a unique value in every record, or a non-key with duplicate values. A secondary index is also a sorted file of records (either fixed-length or variable-length) with two fields. The first field is of the same data type as an indexing field (i.e. a non-ordering field on which the index is built). The second field is either a block pointer or a record pointer. A file may have more than one secondary index.

In this section, we consider two cases of secondary indexes:

- The index access structure constructed on a key field.
- The index access structure constructed on a non-key field.

2.3 Inserting:

The insertion sort, although still $O(n^2)$, works in a slightly different way. It always maintains a sorted sub-list in the lower positions of the list. Each new item is then “inserted” back into the previous sub-list such that the sorted sub-list is one item larger. We begin by assuming that a list with one item (position 00) is already sorted. On each pass, one for each item 1 through $n-1$, the current item is checked against those in the already sorted sub-list. As we look back into the already sorted sub-list, we shift those items that are greater to the right. When we reach a smaller item or the end of the sub-list, the current item can be inserted.

The implementation of insertion sort shows that there are again $n-1$ passes to sort n items. The iteration starts at position 1 and moves through position $n-1$, as these are the items that need to be inserted back into the sorted sub-lists. Line 8 performs the shift operation that moves a value up one position in the list, making room behind it for the insertion. Remember that this is not a complete exchange as was performed in the previous algorithms.

The maximum number of comparisons for an insertion sort is the sum of the first $n-1$ integers. Again, this is $O(n^2)$. However, in the best case, only one comparison needs to be done on each pass. This would be the case for an already sorted list.

One note about shifting versus exchanging is also important. In general, a shift operation requires approximately a third of the processing work of an exchange since only one assignment is performed. In benchmark studies, insertion sort will show very good performance.

2.4 Searching:

When data items are stored in a collection such as a list, we say that they have a linear or sequential relationship. Each data item is stored in a position relative to the others. In Python lists, these relative positions are the index values of the individual items. Since these index values are ordered, it is possible for us to visit them in sequence. This process gives rise to our first searching technique, the sequential search.

The Python implementation for this algorithm is function that needs the list and the item we are looking for and returns a Boolean value as to whether it is present. The Boolean variable found is initialized to False and is assigned the value True if we discover the item in the list.

2.5 Time Complexity:

The time complexity of an algorithm is the total amount of time required by an algorithm to complete its execution. In simple words, every piece of code we write, takes time to execute. The time taken by any piece of code to run is known as the time complexity of that code. The lesser the time complexity, the faster the execution.

If you've programmed a bit before, you're probably wondering how this can be of any use for you because your program was running fine even when you didn't know all of this time complexity stuff, right? I agree with you 100% but there's a catch.

The time for a program to run does not depend solely on efficient of code, it's also dependent on the processing power of a PC. Since time complexity is used to measure the time for algorithms, the type of algorithms you'd use in a small program wouldn't really matter because there's hardly any work being carried out by the processor although when we write code in professional life, the code isn't of 200 or 300 lines.

It's usually longer than a thesis written by a professor and in cases like that, a lot of processor power is being used. If your code is not efficient in terms of data structures, you might find yourself in a rather sticky situation.

CHAPTER 3

IMPLEMENTATION

3.1 Algorithm to build an index:

```
primary_index = {}
secondary_index = {}

for index, row in Index1.csv_df.iterrows():
    my_list = ["Region:" + row['Region'],
              "Country:" + row['Country'],
              "Item Type:" + row['Item Type'],
              "Sales Channel:" + row['Sales Channel'],
              "Order ID:" + str(row['Order ID']),
              "Unit Price:" + str(row['Unit Price'])]

    # Creating primary index.
    Index1.primary_index[index] = my_list

    # Creating secondary index.
    if row['Region'] not in Index1.secondary_index.keys():
        Index1.secondary_index[row['Region']] = [my_list]
    else:
        temp = Index1.secondary_index[row['Region']]
        temp.append(my_list)
```

3.2 Algorithm for insertion:

```
new_data = Index1.get_input()
Index1.csv_df = Index1.csv_df.append(new_data, ignore_index=True)
Index1.csv_df.to_csv(Index1.FILENAME, index=False)
Index1.csv_df = pd.read_csv(Index1.FILENAME)
Index1.create_index()
```

3.3 Algorithm for searching:

```
# Searching using primary index.
for index in range(len(Index1.primary_index)):
    if index == i:
        Index1.prettify_primary(i, Index1.primary_index[i])
        break

# Searching using secondary index.
for region in Index1.secondary_index.keys():
    if region.equals(r):
        list1 = Index1.secondary_index[r]
        Index1.prettify_secondary(r, list1)
        break
```

Time analysis:

Case	Best Case	Worst Case	Average Case
Item is present	1	n	$n+1/2$
Item is not present	n	n	n

3.4 Time Complexity is often measured in terms of :

Big Oh (O) : Worst case running time

In analysis algorithm, Big Oh is often used to describe the worst-case of an algorithm by taking the highest order of a polynomial function and ignoring all the constants value since they aren't too influential for sufficiently large input. So, if an algorithm has running time like $f(n) = 3n + 100$, we can simply state that algorithm has the complexity $O(n)$ which means it always execute at most n procedures(ignoring the constant '100' in between and also the constant '3' being multiplied by n). Thus, we can guarantee that algorithm would not be bad than the worst-case.

Example: prove that $f(n) = 5n^2 + 3n + 2$ has Big Oh $O(n^2)$

Since we know that the highest order of $f(n)$ is 2, we can conclude that $f(n)$ can not have a time complexity greater then that of n^2 , which means it's the worst case running time.

Big Omega (Ω) : Best case running time

Big Omega is often used to describe the best-case running time of an algorithm by choosing the lowest order of the polynomial function and ignoring all the constants.

Example: prove that $f(n) = 5n^2 + 3n$ has Big $\Omega(n)$

We know that the lowest order of the polynomial function $f(n)$ (i.e. 1) is less than n^2 , thus we can conclude that $f(n)$ has a big $\Omega(n)$

Big Theta (Θ) : Both Best and Worst case running time

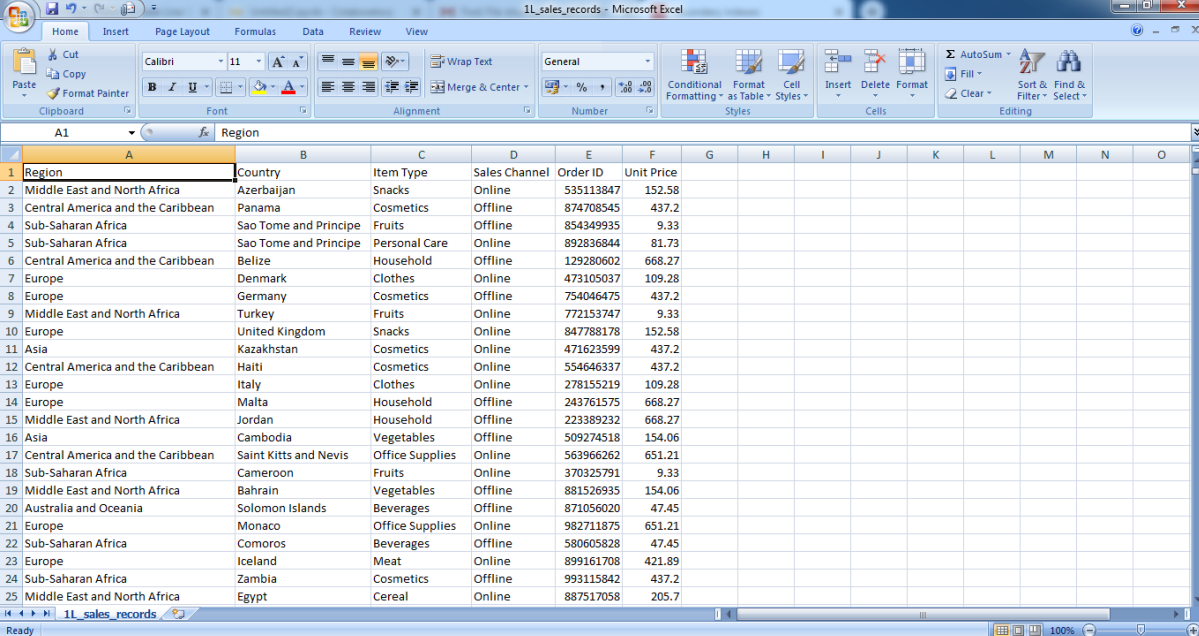
Big Theta describes the running time of an algorithm which is both best case and worst case. This idea might be a big tricky to grasp at first but don't worry, it's not that different from Big Oh or Big Omega, though if you don't completely get the concept of Big Oh and Big Omega, you might want to go through that again before moving onto this. An algorithm has both best and worst case running times. What does that mean? It means that the algorithm is both Big

Oh and Big Omega at the same time. In simpler words, if we consider a polynomial function $f(n) = 3n^3 + 4n$, the Big Theta of the function is going to neither be greater nor smaller than the highest order of the function. It's going to be exactly equal to it, which in this case is going to be n^3 .

CHAPTER 4

RESULTS AND SNAPSHOTS

RESULTS:



Region	Country	Item Type	Sales Channel	Order ID	Unit Price
Middle East and North Africa	Azerbaijan	Snacks	Online	535113847	152.58
Central America and the Caribbean	Panama	Cosmetics	Offline	874708545	437.2
Sub-Saharan Africa	Sao Tome and Principe	Fruits	Offline	854349935	9.33
Sub-Saharan Africa	Sao Tome and Principe	Personal Care	Online	892836844	81.73
Central America and the Caribbean	Belize	Household	Offline	129280602	668.27
Europe	Denmark	Clothes	Online	473105037	109.28
Europe	Germany	Cosmetics	Offline	754046475	437.2
Middle East and North Africa	Turkey	Fruits	Online	772153747	9.33
Europe	United Kingdom	Snacks	Online	847788178	152.58
Asia	Kazakhstan	Cosmetics	Online	471623599	437.2
Central America and the Caribbean	Haiti	Cosmetics	Online	554646337	437.2
Europe	Italy	Clothes	Online	278155219	109.28
Europe	Malta	Household	Offline	243761575	668.27
Middle East and North Africa	Jordan	Household	Offline	223389232	668.27
Asia	Cambodia	Vegetables	Offline	509274518	154.06
Central America and the Caribbean	Saint Kitts and Nevis	Office Supplies	Online	563966262	651.21
Sub-Saharan Africa	Cameroon	Fruits	Online	370325791	9.33
Middle East and North Africa	Bahrain	Vegetables	Offline	881526935	154.06
Australia and Oceania	Solomon Islands	Beverages	Offline	871056020	47.45
Europe	Monaco	Office Supplies	Online	982711875	651.21
Sub-Saharan Africa	Comoros	Beverages	Offline	580605828	47.45
Europe	Iceland	Meat	Online	899161708	421.89
Sub-Saharan Africa	Zambia	Cosmetics	Offline	993115842	437.2
Middle East and North Africa	Egypt	Cereal	Online	887517058	205.7

Fig 4.1: Dataset of Sales Records with Region, Country, Item Type, Sales Channel, Order ID and Unit Price.

```
C:\Users\morab\Anaconda3\envs\bigdata-lab\python.exe D:/IND/Neha/file_structure/Final_Code_INDEXING.py
Welcome: For Indexing
Please wait while data is loading and index is created...
Creating Index Completed.
Time taken to create and build index in seconds: 19.965611696243286

Hit [1] - To Search using Primary Index
Hit [2] - To Search using Secondary Index
Hit [3] - To ADD a NEW row to the csv file
Hit [4] - To DELETE a row from the csv file
Hit [5] - To Display values in the given range of index
Hit [6] - To Exit The File
```

Fig 4.2: Creating and building an index and the time taken to build an index.

```

3
Enter a Start Index:
3456
Enter a Stop Index:
3462
Index: 3456
| Region:Australia and Oceania | Country:Samoa | Item Type:Clothes | Sales Channel:Online | Order ID:899043262 | Unit Price:109.28 |
Index: 3457
| Region:Sub-Saharan Africa | Country:Burundi | Item Type:Fruits | Sales Channel:Online | Order ID:764426839 | Unit Price:9.33 |
Index: 3458
| Region:Asia | Country:Sri Lanka | Item Type:Beverages | Sales Channel:Online | Order ID:857405936 | Unit Price:47.45 |
Index: 3459
| Region:Middle East and North Africa | Country:Jordan | Item Type:Cereal | Sales Channel:Offline | Order ID:317091882 | Unit Price:205.7 |
Index: 3460
| Region:Middle East and North Africa | Country:United Arab Emirates | Item Type:Cosmetics | Sales Channel:Offline | Order ID:152908504 | Unit Price:437.2 |
Index: 3461
| Region:Europe | Country:Vatican City | Item Type:Fruits | Sales Channel:Offline | Order ID:994592130 | Unit Price:9.33 |
Index: 3462
| Region:Central America and the Caribbean | Country:Grenada | Item Type:Snacks | Sales Channel:Offline | Order ID:842191588 | Unit Price:152.58 |

```

Fig 4.3: Records has been displayed with range limit.

```

3
Enter the Region: Europe
Enter Country Name: Denmark
Enter Item Type: Kurkure
Enter Sales Channel: Offline
Enter Order ID: 123456789
Enter Unit Price: 10.0
Please wait while data is loading and index is created...
Creating Index Completed.
Time taken to create and build index in seconds: 20.708625555038452

Time taken to insert in a file in milli seconds: 142014.32568359375

Hit [1] - To Search using Primary Index
Hit [2] - To Search using Secondary Index
Hit [3] - To ADD a NEW row to the csv file
Hit [4] - To DELETE a row from the csv file
Hit [5] - To Display values in the given range of index
Hit [6] - To Exit The File

```

Fig 4.4: Here we are inserting the data into the file, insertion is successful. The time taken for insertion is calculated and displayed.

Europe	Poland	Meat	Offline	110449349	421.89
Sub-Saharan Africa	Comoros	Clothes	Online	193128764	109.28
Middle East and North Africa	Kuwait	Cosmetics	Online	701597058	437.2
Sub-Saharan Africa	Tanzania	Cosmetics	Offline	423403060	437.2
Europe	Denmark	Kurkure	Offline	123456789	10

Fig 4.5: The data has been successfully inserted into the csv file

```

Hit [1] - To Search using Primary Index
Hit [2] - To Search using Secondary Index
Hit [3] - To ADD a NEW row to the csv file
Hit [4] - To DELETE a row from the csv file
Hit [5] - To Display values in the given range of index
Hit [6] - To Exit The File

1
Enter a Search Index:
25000
Index: 25000
| Region:North America | Country:Greenland | Item Type:Personal Care | Sales Channel:Offline | Order ID:434001863 | Unit Price:81.73 |

Time taken to search using primary index in milli seconds: 3.989501953125

```

Fig 4.6: Searching a particular data in a csv file using primary index and the search is successful.

```

| Region:Asia | Country:Maldives | Item Type:Cereal | Sales Channel:Online | Order ID:783513438 | Unit Price:203.7 |
| Region:Asia | Country:Taiwan | Item Type:Personal Care | Sales Channel:Offline | Order ID:936946094 | Unit Price:81.73 |
| Region:Asia | Country:Malaysia | Item Type:Cosmetics | Sales Channel:Offline | Order ID:592965018 | Unit Price:437.2 |
| Region:Asia | Country:South Korea | Item Type:Clothes | Sales Channel:Online | Order ID:779845893 | Unit Price:109.28 |
| Region:Asia | Country:Camodia | Item Type:Fruits | Sales Channel:Online | Order ID:777176129 | Unit Price:9.33 |
| Region:Asia | Country:North Korea | Item Type:Fruits | Sales Channel:Online | Order ID:388618028 | Unit Price:9.33 |
| Region:Asia | Country:Maldives | Item Type:Snacks | Sales Channel:Offline | Order ID:296343600 | Unit Price:152.58 |
| Region:Asia | Country:Vietnam | Item Type:Baby Food | Sales Channel:Offline | Order ID:573824346 | Unit Price:255.28 |

Time taken to search using secondary index in milli seconds: 273.267333984375

Hit [1] - To Search using Primary Index
Hit [2] - To Search using Secondary Index
Hit [3] - To ADD a NEW row to the csv file
Hit [4] - To DELETE a row from the csv file
Hit [5] - To Display values in the given range of index
Hit [6] - To Exit The File

```

Fig 4.7: Searching data in a csv file using secondary index and the search is successful.

```

Hit [1] - To Search using Primary Index
Hit [2] - To Search using Secondary Index
Hit [3] - To ADD a NEW row to the csv file
Hit [4] - To DELETE a row from the csv file
Hit [5] - To Display values in the given range of index
Hit [6] - To Exit The File

4
Enter a Index to Delete:
1000
Please wait while data is loading and index is created...
Creating Index Completed.
Time taken to create and build index in seconds:  20.227909326553345

Time taken to delete a file in milli seconds:  21118.527587890625

```

Fig 4.8: Here we can see deletion of a record has been successful.

TIME ANALYSIS:

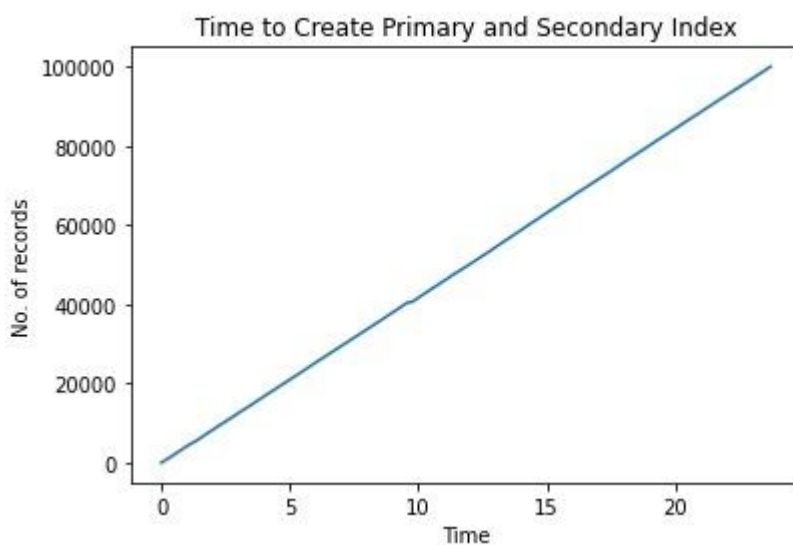


Fig 4.8: In the above figure we can know the time analysis graph to build an index for the given dataset.

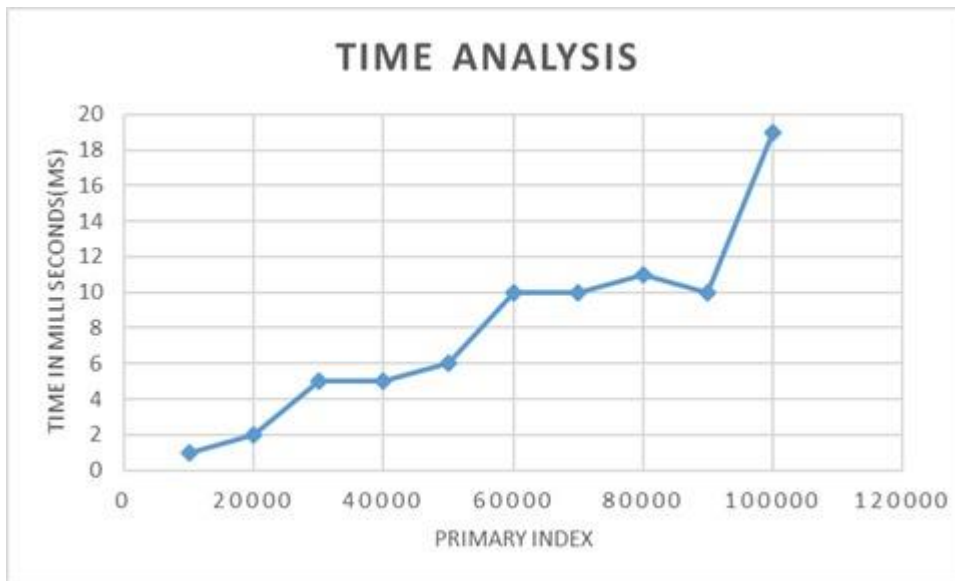


Fig 4.9: In the above figure we can know the time analysis graph for searching a data in the dataset using primary index

CONCLUSION

Conclusion

- Users search in a hurry for information to help them and give up after two or three tries.
- An Indexes form an important part of designing, creating and testing information.
- Index can point the way in harmony with user expectations or not.
- Indexing is an interactive analysis and creative process throughout the entire documentation.
- Helps to make a row unique or without duplicates.
- If index is set to full-text index, then we can search against large string values.
- It supports different type of indexes like primary key index, unique index, normal index and full-text index. Indexes help to speed up the retrieval of data from database.

REFERENCES

1. GeeksforGeeks
2. Tutorials Point
3. Google