

**Visvesvaraya Technological University
Belagavi-590018, Karnataka**



A Mini Project Report on

“Movies/Series Search tool with Indexing”

Submitted in partial fulfilment of the requirement for the
FILE STRUCTURES LAB [18ISL68]

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

Submitted by

SRICHARAN B.S [1JT18IS055]

Under the guidance of
Mr. Vadiraja A
Assistant Professor, Dept of ISE



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

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



CERTIFICATE

Certified that the mini project work entitled “**Movies/Series Search tool with Indexing**” carried out by **Sricharan B.S [1JT18IS055]** bonafide student of Jyothy Institute of Technology, in partial fulfilment 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 mini project report has been approved as it satisfies the academic requirements in respect of Mini Project work prescribed for the said Degree.

Mr. Vadiraja A

Guide, Asst. Professor
Dept. Of ISE

External Viva Examiner

- 1.
- 2.

Dr.Harshwardhan Tiwari

Assoc. Professor and HOD
Dept. Of ISE

Signature with Date :

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. Harshwardhan Tiwari, Professor and Head** of Information Science and Engineering Department for providing for his valuable support.

We would like to thank our guides **Mr.Vadiraja A, Asst. Prof.** for her keen interest and guidance in preparing this work.

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

Sricharan B.S [1JT18IS055]

ABSTRACT

This project is to be implemented in python using libraries like csv, pandas etc as per the requirements. The raw data is fetched from IMDB website and dataset is obtained a .tsv file and further is converted to .csv file for the operational purposes.

Data Obtaining, understanding and Cleaning using python code snippets were the prerequisites done before carrying out operations like searching ,inserting ,deleting and indexing .These prerequisites are done according to the fields required to obtain in accessible dataset.

This terminal based search tool includes the **selective searching** where the user can select an attribute to search a particular movie/series by giving its title or released year or combination of both or its genre or its language as a selected input and also have the option to **save** the obtained result to a **.csv** file.

This tool project shows the implementation of **primary and secondary key indexing** and have options to search ,insert and delete based on the primary and secondary key obtained .

The obtained results are compared into graphs, time and space constraints to check the efficiency and accessibility of this particular Movie/TV-Series finder tool.

.

TABLE OF CONTENTS

| SLNO | TITLE | PAGE NO. |
|-------------|--|---------------------------|
| 1 | INTRODUCTION 1.1 Introduction to file structures 1.2 Introduction to python 1.3 Introduction to indexing 1.4 Importance of Indexing 1.5 Objectives and scope of importance of work | 1 1 2 3 3 |
| 2 | IMPLEMENTATION 2.1 Indexing Algorithm 2.2 Linear search Algorithm | 4 5 |
| 3 | RESULTS AND SNAPSHOTS | 6-15 |
| 4 | REFERENCES AND CONCLUSIONS | 16 |

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.

On the whole 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 actually 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 particular logical structure is stored on, or in it. Typically for every unit of physical store there will be a number of 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 Python

Python is a dynamic, interpreted (byte code-compiled) language. There are no type declarations of variables, parameters, functions, or methods in source code. This makes the code short and flexible, and you lose the compile-time type checking of the source code. Python tracks the types of all values at runtime and flags code that does not make sense as it runs.

An excellent way to see how Python code works is to run the Python interpreter and type code right into it. If you ever have a question like, "What happens if I add an int to a list?" Just typing it into the Python interpreter is a fast and likely the best way to see what happens.

Python source files use the ".py" extension and are called "modules." One unusual Python feature is that the whitespace indentation of a piece of code affects its meaning. A logical block of statements such as the ones that make up a function should all have the same indentation, set in from the indentation of their parent function or "if" or whatever. If one of the lines in a group has a different indentation, it is flagged as a syntax error.

1.3 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.

The second column is the Data Reference or Pointer which contains a set of pointers holding the address of the disk block where that particular key value can be found.

Primary Indexing:

If the index is created on the primary key then it is called as primary indexing. Since these primary keys are unique to each record and it has 1:1 relation between the records, it is much easier to fetch the record using primary indexing. There are two types of primary indexing – Dense index and Sparse index.

Dense index- indexing is created based on primary as well as on columns which means user can search on any column. The dense index addresses quick search on any search, the space used for index and address can be queried based on even columns. Hence index on all search key columns are stored. This becomes overhead in the memory. Therefore more space is consumed to store the indexes and the record size increases.

Sparse index- in this method of indexing, range of index columns store the same data block address and when the data is to be retrieved, the block address will be fetched linearly till we get the requested data.

Secondary Indexing

In sparse indexing as the table size grows, the (index, address) mapping file size also grows which results in slower fetching of address.

In secondary indexing method, another level of indexing is introduced to reduce the mapping size. That means initially huge range for columns are selected so that the first level of mapping is small. Then each range is divided into smaller ranges. First level of mapping is stored in the primary memory so that address can be fetched faster. Secondary level of mapping and the actual data are stored in the secondary memory.

1.4 Importance of Indexing

- **Easy location** – Indexing points out the required records or file and facilitates easy location. Saves time and effort – Indexing gives the ready reference to the records and saves time and efforts of the office.
- **Efficiency** – Indexing helps to find out the records easily and quickly which enhances the efficiency.
- **Reduce costs** – Indexing helps to reduce the cost by saving time and efforts.
- **Cross Reference** – A particular record can be maintained through two ways.

Indexing facilitates to find out such records through cross reference. Index is not only necessary to large office but also necessary for small office. When a large number of files are maintained, the necessity of maintaining index is increased. Indexing increases the utility of filing by providing an easy reference to files. The very purpose of maintaining index is that it is easy and location of filing is faster.

1.5 Objectives and scope of importance of work:

The main objective of this mini project was to develop a movie/series search tool by using file structure domains like primary/secondary indexing and linear searching.

First to obtain the feasible -accessible dataset suitable to perform the above file structure domain, a data analysis was done which included the analysis of data, which further included the inspection, observation, understanding, merging and cleaning of the raw dataset obtained from IMDB interface and further convert that tsv interface to feasible dataset csv file.

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.

So, to meet the objectives, A feasible dataset as mentioned above was used to perform tasks like primary indexing, searching /inserting/deleting using primary index, secondary indexing, searching /inserting/deleting using secondary indexing, and a user menu terminal-based selective searching where the user can select between various fields of movies/series domains to search from i.e., domains like title, year, genre, language, director, actor, runtime etc. The tool also provides an option to save the obtained results from search to a csv file whenever the result count exceeds more than 1.

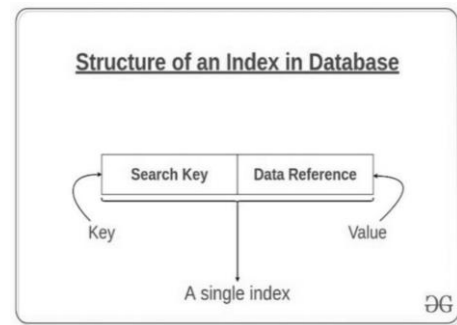
Further to describe the efficiency of the linear search algorithm and the primary/secondary indexing it has used time and space constraints – time constrain is the number of msec time taken vs the no of records and space constraint is the size of csv file generated during the searching process

IMPLEMENTATION

2.1 Basic operations on Indexing:

In this section, we present the details of the operations of indexing:

- Entering the details.
- Searching.
- Deleting.
- Build Index
- Modify



Algorithm:

Step 1: Accessing a particular dataset.

Step 2: Create 2 new index files, one for primary indexing and the other for secondary indexing.

Step 3: The first column is the search key that contains a copy of primary key or candidate key of the table.

Step 4: The second column is the pointer which contains a set of pointers holding the address of the disk block where that particular key value is found.

Step 5: Record insertion - This consists of appending the data file and inserting a new record. The rearrangement of the index consists of sliding down the records with keys larger than the inserted key and then placing new record in the opened space.

Step 6: Record deletion-This should use the techniques for reclaiming space in files when deleting from the data file. We must delete the corresponding entry from the index. Shift all records with keys larger than key of the deleted record to the previous position in memory or make the index entry as deleted.

Step 7: In our record file we built an index for 'ID' which is primary key and there is 'S_ID' as secondary key.

Step 8: Record deletion in secondary key: Deleting a record implies removing all the references to the record in primary index and in all secondary indexes. When accessing the file through secondary key, the primary indexed file will be checked and a deleted record can be identified.

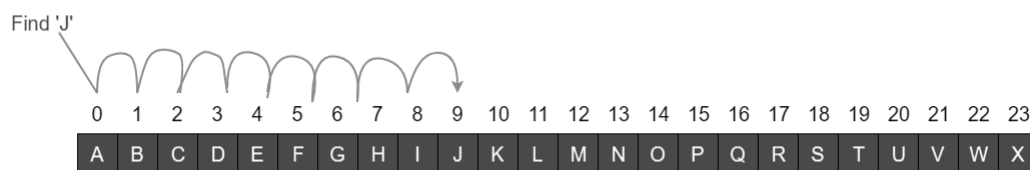
Step 9: It allows binary search to obtain a keyed access to a record in variable length record file.

Step 10: Time taken for dataset has been calculated for each functionality.

2.2 Linear search algorithm

To perform the selective searching and primary/secondary index searching, the linear search algorithm was used

Linear search is a method of finding elements within a list. It is also called a sequential search. It is the simplest searching algorithm because it searches the desired element in a sequential manner. It compares each and every element with the value that we are searching for. If both are matched, the element is found, and the algorithm returns the key's index position.



Algorithm:

There is list of n elements and key value to be searched.

```
LinearSearch(list, key)
for each item in the list
    if item == value
        return its index position
return -1
```

A simple approach is to do linear search, i.e

Start from the leftmost element of arr[] and one by one compare x with each element of arr[]
If x matches with an element, return the index.
If x doesn't match with any of elements, return -1.

Linear Search Complexities

Time Complexity: $O(n)$

Space Complexity: $O(1)$

RESULTS AND SNAPSHOTS

HOME MENU :

```
sricharanvardhan@LAPTOP-UN748RLJ MINGW64 ~/Desktop/Fs mini project
$ python main.py

-----WELCOME TO MOVIE-TVSERIES SEARCH TOOL-----

PRESS 1 FOR MOVIES
PRESS 2 FOR TV SERIES
PRESS 0 FOR EXIT
1

PRESS 1 TO SEARCH AND OBTAIN MOVIES
PRESS 2 TO IMPLEMENT PRIMARY KEY INDEXING ON MOVIES
PRESS 3 TO SEARCH MOVIES USING PRIMARY KEY
PRESS 4 TO IMPLEMENT SECONDARY KEY INDEXING ON MOVIES
PRESS 5 TO SEARCH MOVIES USING SECONDARY KEY
PRESS 6 TO INSERT A MOVIE TO DATASET
PRESS 7 TO DELETE A MOVIE FROM DATASET USING PRIMARY KEY
PRESS 8 TO DELETE A MOVIE FROM DATASET USING SECONDARY KEY

PRESS 0 TO EXIT
0










Exiting...

PRESS 1 FOR MOVIES
PRESS 2 FOR TV SERIES
PRESS 0 FOR EXIT
2

PRESS 1 TO SEARCH AND OBTAIN TV-SERIES
PRESS 2 TO IMPLEMENT PRIMARY KEY INDEXING ON TV-SERIES
PRESS 3 TO SEARCH TV-SERIES USING PRIMARY KEY
PRESS 4 TO IMPLEMENT SECONDARY KEY INDEXING ON TV-SERIES
PRESS 5 TO SEARCH TV-SERIES USING SECONDARY KEY
PRESS 6 TO INSERT A TV-SERIES TO DATASET
PRESS 7 TO DELETE A TV-SERIES FROM DATASET USING PRIMARY KEY
PRESS 8 TO DELETE A TV-SERIES FROM DATASET USING SECONDARY KEY

PRESS 0 TO EXIT
```

DATA SET CREATION :

| mini project > datacreation | | | | |
|---|------------------|------------------------|-------------|--|
| Name | Date modified | Type | Size | |
|  finalseries.csv | 11-08-2021 22:45 | Microsoft Excel Com... | 11,047 KB | |
|  finalsmovies.csv | 11-08-2021 22:44 | Microsoft Excel Com... | 24,283 KB | |
|  mergescsv.py | 11-08-2021 23:18 | Python File | 1 KB | |
|  movies.csv | 29-05-2021 18:00 | Microsoft Excel Com... | 25,663 KB | |
|  ratings1.csv | 29-05-2021 19:52 | Microsoft Excel Com... | 20,399 KB | |
|  title.basics.tsv | 24-05-2021 18:08 | TSV File | 6,61,448 KB | |
|  title.ratings.tsv | 29-05-2021 19:34 | TSV File | 19,272 KB | |
|  tsvtocsv.py | 11-08-2021 23:19 | Python File | 1 KB | |
|  tvseries.csv | 24-05-2021 19:21 | Microsoft Excel Com... | 10,000 KB | |

Obtaining raw dataset from IMDB interface in tsv format and converting /merging/ segregating and obtaining two feasible datasets one for movie and one for series . The highlighted part shows the reduce of size in kb when the operation is carried out.

MOVIE SEARCH-MENU :

```
MINGW64/c/Users/sricharanvardhan/Desktop/Fs mini project
sricharanvardhan@LAPTOP-UN748RLJ MINGW64 ~/Desktop/Fs mini project
$ python main.py

-----WELCOME TO MOVIE-TVSERIES SEARCH TOOL-----

PRESS 1 FOR MOVIES
PRESS 2 FOR TV SERIES
PRESS 0 FOR EXIT
1

PRESS 1 TO SEARCH AND OBTAIN MOVIES
PRESS 2 TO IMPLEMENT PRIMARY KEY INDEXING ON MOVIES
PRESS 3 TO SEARCH MOVIES USING PRIMARY KEY
PRESS 4 TO IMPLEMENT SECONDARY KEY INDEXING ON MOVIES
PRESS 5 TO SEARCH MOVIES USING SECONDARY KEY
PRESS 6 TO INSERT A MOVIE TO DATASET
PRESS 7 TO DELETE A MOVIE FROM DATASET USING PRIMARY KEY
PRESS 8 TO DELETE A MOVIE FROM DATASET USING SECONDARY KEY

PRESS 0 TO EXIT
1

PRESS 1 to search a particular movie by its title
PRESS 2 to obtain the movie according to year
PRESS 3 to obtain all the movies by a particular director
PRESS 4 to obtain all the movies by a particular actor
PRESS 5 to obtain all the movies by a particular language
PRESS 6 to obtain all the movies by a particular genre
PRESS 7 to obtain by rating:

PRESS 0 to Exit
```

SERIES SEARCH-MENU :

```
MINGW64/c/Users/sricharanvardhan/Desktop/Fs mini project
sricharanvardhan@LAPTOP-UN748RLJ MINGW64 ~/Desktop/Fs mini project
$ python main.py

-----WELCOME TO MOVIE-TVSERIES SEARCH TOOL-----

PRESS 1 FOR MOVIES
PRESS 2 FOR TV SERIES
PRESS 0 FOR EXIT
2

PRESS 1 TO SEARCH AND OBTAIN TV-SERIES
PRESS 2 TO IMPLEMENT PRIMARY KEY INDEXING ON TV-SERIES
PRESS 3 TO SEARCH TV-SERIES USING PRIMARY KEY
PRESS 4 TO IMPLEMENT SECONDARY KEY INDEXING ON TV-SERIES
PRESS 5 TO SEARCH TV-SERIES USING SECONDARY KEY
PRESS 6 TO INSERT A TV-SERIES TO DATASET
PRESS 7 TO DELETE A TV-SERIES FROM DATASET USING PRIMARY KEY
PRESS 8 TO DELETE A TV-SERIES FROM DATASET USING SECONDARY KEY

PRESS 0 TO EXIT
1

PRESS 1 to search a particular series by its title
PRESS 2 to search the series according to start year
PRESS 3 to search the series by its genre
PRESS 4 to search by rating:

PRESS 0 to Exit
```

SEARCHING A MOVIE BY ITS TITLE :

```
MINGW64/c/Users/sricharanvardhan/Desktop/Fs mini project

PRESS 0 TO EXIT
1

PRESS 1 to search a particular movie by its title
PRESS 2 to obtain the movie according to year
PRESS 3 to obtain all the movies by a particular director
PRESS 4 to obtain all the movies by a particular actor
PRESS 5 to obtain all the movies by a particular language
PRESS 6 to obtain all the movies by a particular genre
PRESS 7 to obtain by rating:

PRESS 0 to Exit
1
Enter the name of movie you want to find :pulp Fiction

tconst : tt0110912

movie name : Pulp Fiction

Released Year : 1994

Genre : Crime, Drama

Language : English, Spanish, French

Director : Quentin Tarantino

Actors : Tim Roth, Amanda Plummer, Laura Lovelace, John Travolta, Samuel L. Jackson, Phil LaMarr, Frank Whaley, Burr Steers, Bruce Willis, Ving Rhames, Paul Calderon, Brona
gh Gallagher, Rosanna Arquette, Eric Stoltz, Uma Thurman

Rating : 8.9

\N indicates that the data has not been loaded

Matches found :1
Completed the task in 0.71 seconds
```

TO OBTAIN ALL MOVIES RELEASED BETWEEN 2 YEARS :

```
PRESS 1 to obtain all movie according to one particular start year
PRESS 2 to obtain the movie by its title and its start year
PRESS 3 to obtain all the movie present between two years of your choice
PRESS 4 to obtain all the movie started after a particular year
PRESS 5 to obtain all the movie started before a particular year

PRESS 0 to Exit
3
Enter the from year to start with:2019
Enter the to year :2020

Director : Laura Jou

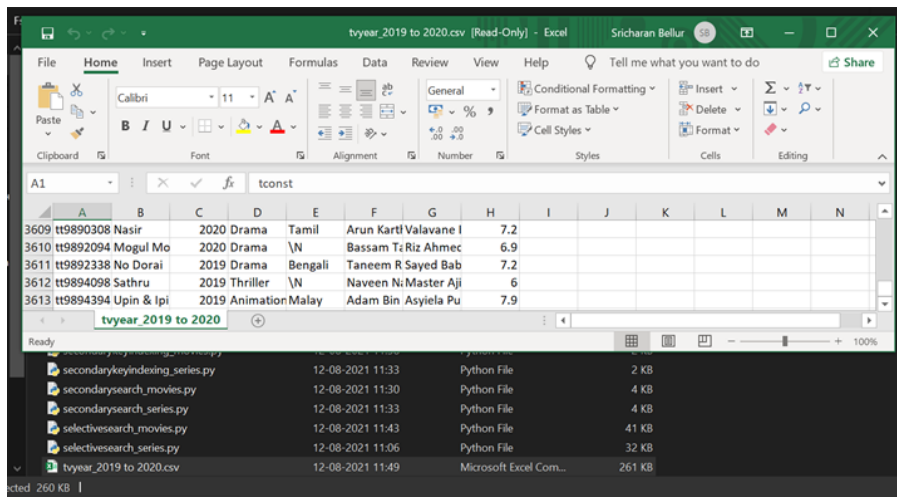
Actors : Maria Morera Colomer, Biel Rossell Pelfort, Isaac Alcayde, Lluís Altés, Joan Amargós, Pepo Blasco, Cesc Casanovas, Oriol Cervera, Pau Escobar, Jordi
Figueras, Arés Fuster, Judit Martín, Martí Múrcia, Mariona Pagès, Francesca Piñón

Rating : 6.7

\N indicates that the data has not been loaded

Matches found :3630
Do you want to save it in a csv file? [1/0]
1
Check in the current directory for the csv file
```

The above search result is obtained and the .csv file is stored as per the user option.
Below it's a pic of the csv file obtained and as u can see its stored in the same directory



TO OBTAIN ALL THE MOVIES BY A PARTICULAR DIRECTOR:

```

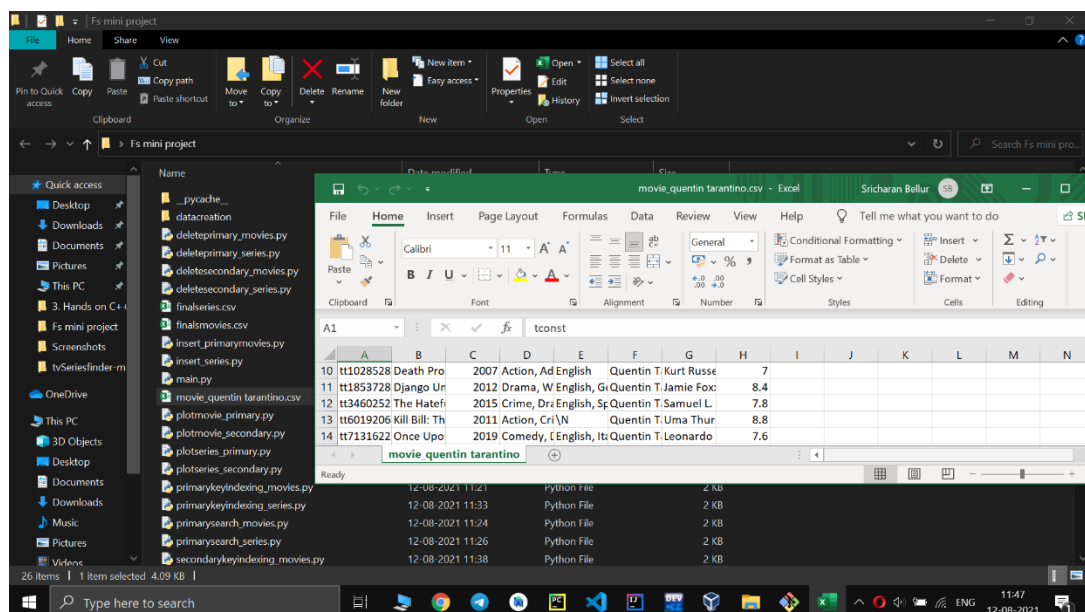
PRESS 3 to obtain all the movies by a particular director
PRESS 4 to obtain all the movies by a particular actor
PRESS 5 to obtain all the movies by a particular language
PRESS 6 to obtain all the movies by a particular genre
PRESS 7 to obtain by rating:

PRESS 0 to Exit
3
Enter the name of director to search : quentin tarantino

Movie name : Once Upon a Time... in Hollywood
Released Year : 2019
Genre : Comedy, Drama
Language : English, Italian, Spanish, German
Director : Quentin Tarantino
Actors : Leonardo DiCaprio, Brad Pitt, Margot Robbie, Emile Hirsch, Margaret Qualley, Timothy Olyphant, Julia Butters, Austin Butler, Dakota Fanning, Bruce D
ern, Mike Moh, Luke Perry, Damian Lewis, Al Pacino, Nicholas Hammond
Rating : 7.6
\N indicates that the data has not been loaded

Matches found :13
Completed the task in 0.81 seconds
Do you want to save it in a csv file? [Y/N]
Y
Check in the current directory for the csv file
Completed the task in 0.50 seconds
  
```

The above search result is obtained and the .csv file is stored as per the user option. Below it's a pic of the csv file obtained and as u can see its stored in the same directory



OBTAIN ALL THE MOVIES BY A PARTICULAR ACTOR:

```
PRESS 4 to obtain all the movies by a particular actor
PRESS 5 to obtain all the movies by a particular language
PRESS 6 to obtain all the movies by a particular genre
PRESS 7 to obtain by rating:

PRESS 0 to Exit
4
Enter the name of actor to search : morgan freeman

Tconst : tt0080474

movie name : Brubaker

Released Year : 1980

Genre : Crime, Drama

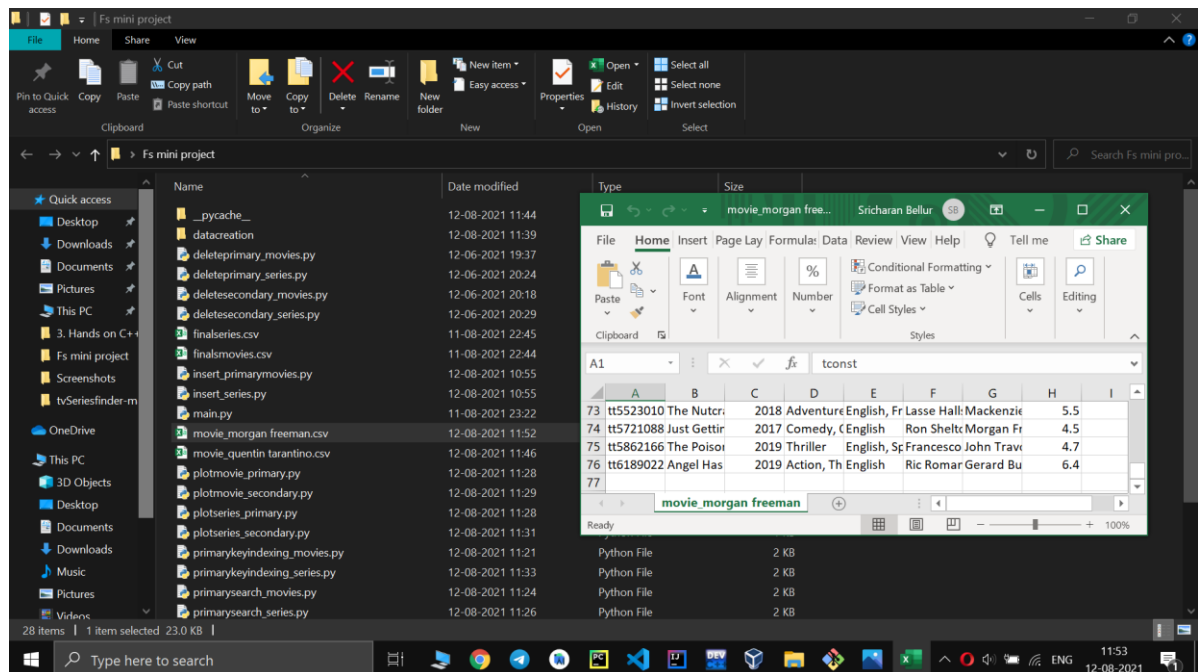
Actors : Gerard Butler, Frederick Schmidt, Danny Huston, Rocci Williams, Piper Perabo, Harry Ditson, Linda John-Pierre, Lance Reddick, Ori Pfeffer, Morgan Freeman, Jasmine Hyde, Ian Porter, Laurel Lefkow, Michael Landes, Mark Arnold

Rating : 6.4

\\N indicates that the data has not been loaded

Matches found :75
Completed the task in 26.25 seconds
Do you want to save it in a csv file? [1/0]
1
Check in the current directory for the csv file
Completed the task in 26.61 seconds
```

The above search result is obtained and the .csv file is stored as per the user option. Below it's a pic of the csv file obtained and as u can see its stored in the same directory



OBTAIN ALL THE MOVIES BY A PARTICULAR ACTOR:

```
PRESS 3 to obtain all the movies by a particular director
PRESS 4 to obtain all the movies by a particular actor
PRESS 5 to obtain all the movies by a particular language
PRESS 6 to obtain all the movies by a particular genre
PRESS 7 to obtain by rating:

PRESS 0 to Exit
5
Enter the name of language to search : kannada

Language : Kannada

Director : Navarasan

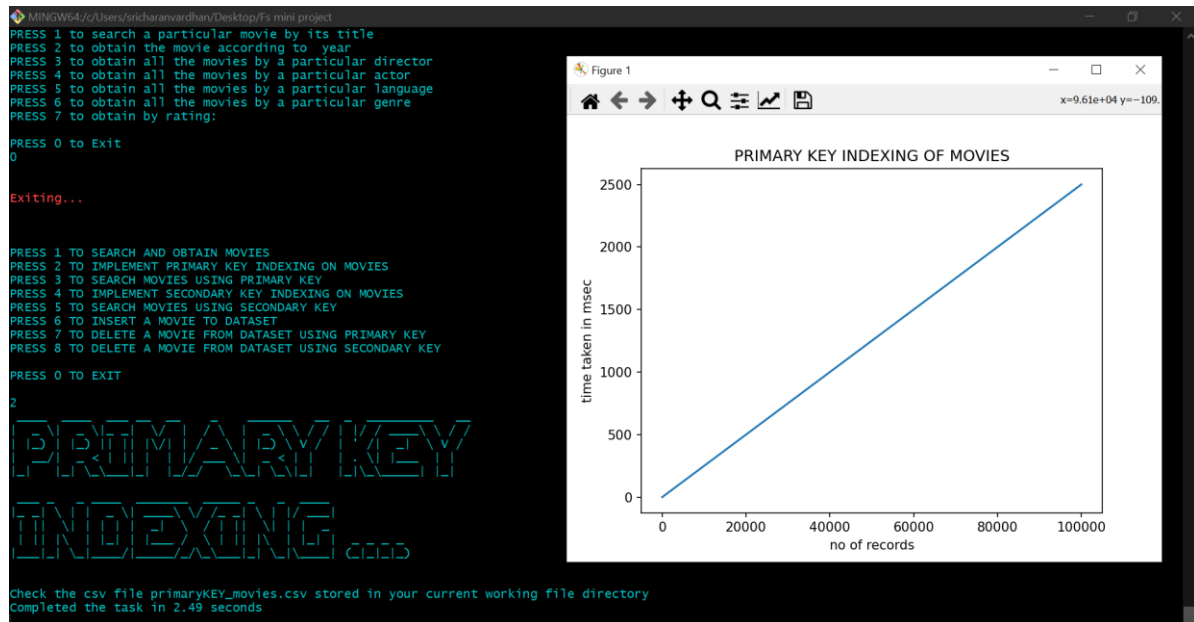
Actors : Radhika Kumaraswamy, Adya, Amarnath, Anjana, Lalana R. Arya, Raj Bahadur, Kavya Gowda, Ravi Gowda, Tanu Gowda, Karthik, Krishna Murthi Kawthar, Kempegowda, Sadhu Kokila, Honnavalli Krishna, Naveen Krishna

Rating : 8.4

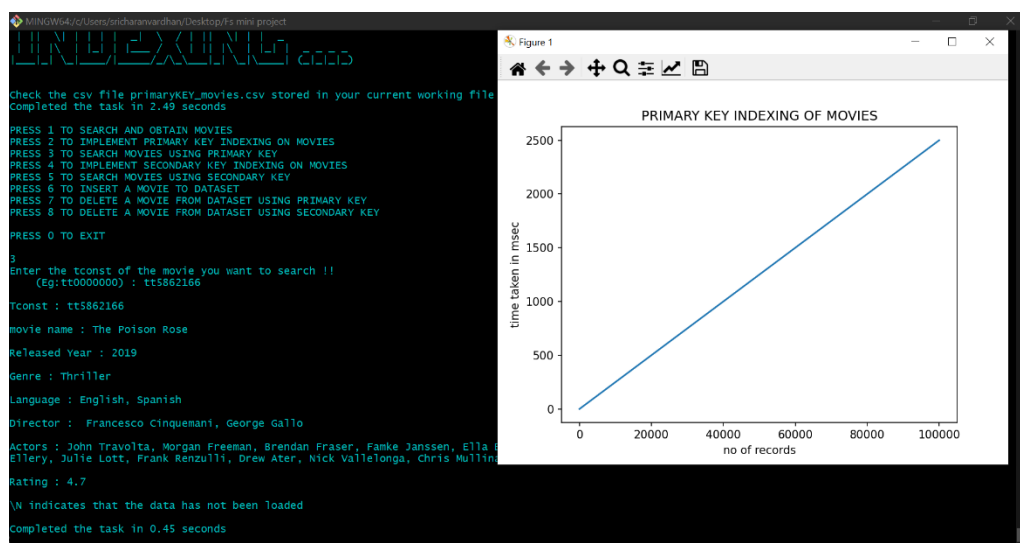
\\N indicates that the data has not been loaded

Matches found :240
Completed the task in 10.10 seconds
Do you want to save it in a csv file? [1/0]
1
Check in the current directory for the csv file
Completed the task in 0.57 seconds
```

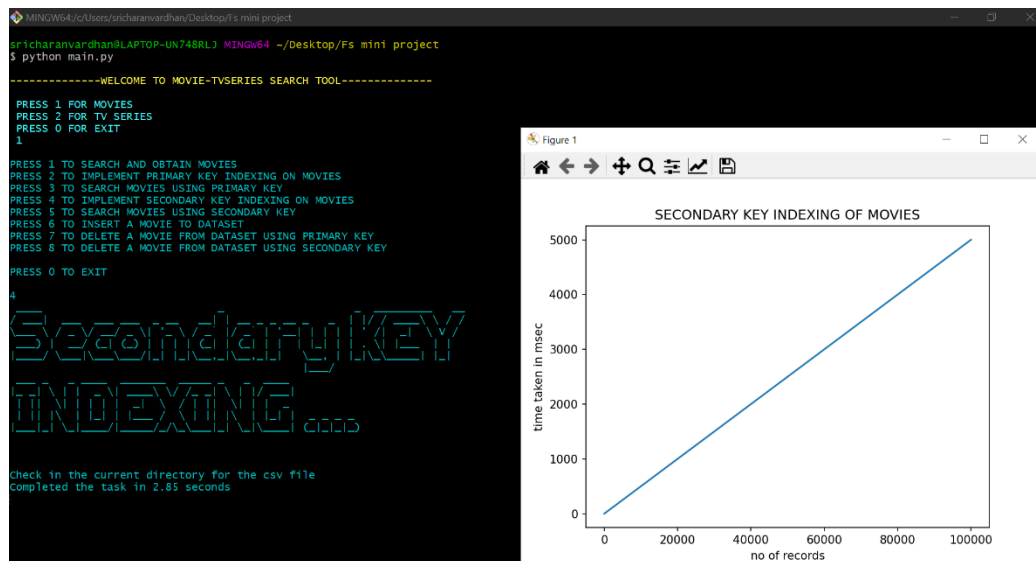
PRIMARY KEY INDEXING AND SEARCHING :



SEARCHING:



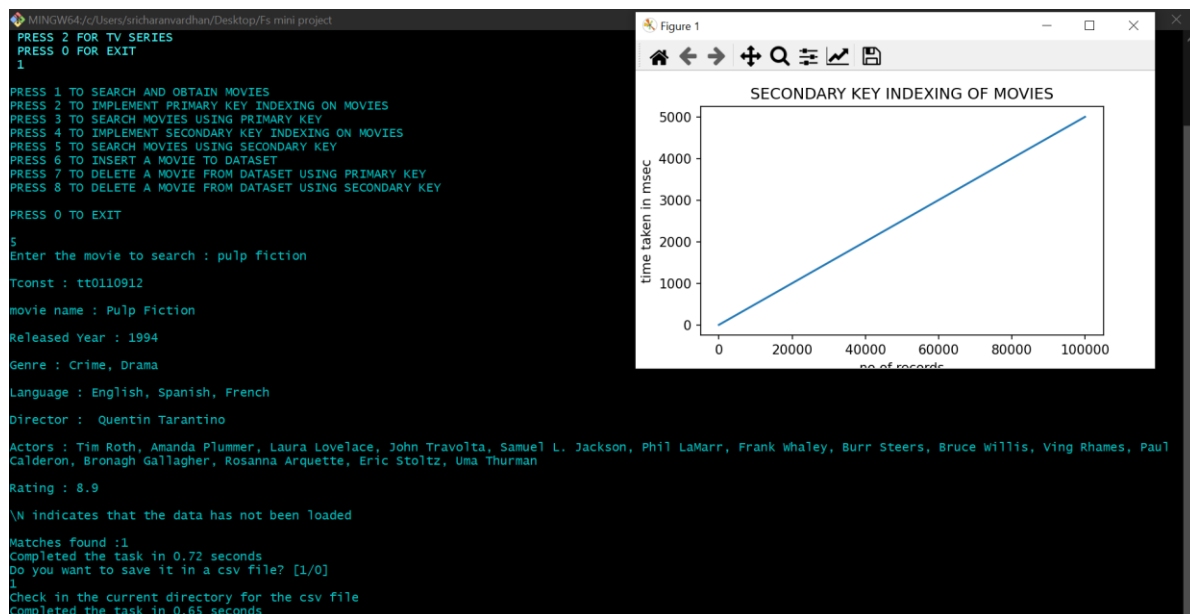
SECONDARY SEARCHING KEY INDEXING AND SEARCHING:



The file explorer shows the 'Fs mini project' directory containing various CSV and Python files. The Excel spreadsheet, titled 'secondaryKEY_movies', displays the results of the indexing process. The data is organized into columns for ID, Title, and other details.

| ID | Title |
|-------|---|
| 85846 | 24862871 Jessie |
| 85847 | 24863169 La reina de los lagartos |
| 85848 | 24863332 Enemy Lines |
| 85849 | 24863661 Ottam |
| 85850 | 24863874 Pengallia |
| 85851 | 24864011 Manoharam |
| 85852 | 24864296 Le lion |
| 85853 | 24864578 De Beentjes van Sint-Hildegard |
| 85854 | 24864930 Padmavyuhathile Abhimanyu |
| 85855 | 24865069 Sokagin Ātucuklari |
| 85856 | 24865382 La vida sense la Sara Amat |

SEARCHING



INSERTION:

The terminal window shows a series of commands and prompts for inserting a movie into a dataset. The user enters the title 'jit movies', the year '2021', the genre 'Horror', the language 'Kannada', and the director 'Principal'. The Excel spreadsheet shows the resulting data entry in the 'finalmovies.csv' file.

| A | B | C | D | E | F | G | H |
|-------|-----------|-------------|------|--------|---------|--------------------|-----|
| 85856 | tt9914942 | La vida sei | 2019 | Drama | Catalan | Laura Jou Maria Mo | 6.7 |
| 85857 | tt1111111 | jit movies | 2021 | Horror | Kannada | Principal | 10 |

DELETION(PRIMARY KEY):

The terminal window shows a series of commands and prompts for deleting a movie from a dataset. The user enters the title 'The Married Virgin', the year '1918', the genre 'Drama', the language 'English', and the director 'Joseph Maxwell'. The Excel spreadsheet shows the resulting data entry in the 'finalmovies deleted_primary' file.

| A | B | C | D | E | F | G | H | I |
|-----|-----------|-----------|------|---------|---------|-------------|-------------|------------|
| 127 | tt0009153 | Hell Bent | 1918 | Western | W | John Ford | Harry Carr | 6.1 |
| 128 | tt0009241 | Johanna E | 1918 | Comedy | F | English | William D | Mary Pickl |
| 129 | tt0009326 | M'Liss | 1918 | Comedy | E | English | Marshall | Mary Pickl |
| 130 | tt0009369 | Mickey | 1918 | Comedy | E | English | F. Richard | Mabel Noi |
| 131 | tt0009440 | Old Wives | 1918 | Drama | English | Cecil B. De | Elliott Dex | 6.4 |

DELETION(SECONDARY KEY):

The terminal window shows a series of commands and prompts for deleting a movie from a dataset. The user enters the title 'Harry Potter and the Sorcerer's Stone', the year '2001', the genre 'Adventure', the language 'English', and the director 'Chris Columbus'. The terminal shows the resulting data entry in the 'finalmovies deleted_primary' file.

| A | B | C | D | E | F | G | H | I |
|-----|-----------|-----------|------|---------|---------|-------------|-------------|------------|
| 127 | tt0009153 | Hell Bent | 1918 | Western | W | John Ford | Harry Carr | 6.1 |
| 128 | tt0009241 | Johanna E | 1918 | Comedy | F | English | William D | Mary Pickl |
| 129 | tt0009326 | M'Liss | 1918 | Comedy | E | English | Marshall | Mary Pickl |
| 130 | tt0009369 | Mickey | 1918 | Comedy | E | English | F. Richard | Mabel Noi |
| 131 | tt0009440 | Old Wives | 1918 | Drama | English | Cecil B. De | Elliott Dex | 6.4 |

SEARCHING A SERIES BY ITS TITLE:

```
PRESS 1 TO SEARCH AND OBTAIN TV-SERIES
PRESS 2 TO IMPLEMENT PRIMARY KEY INDEXING ON TV-SERIES
PRESS 3 TO SEARCH TV-SERIES USING PRIMARY KEY
PRESS 4 TO IMPLEMENT SECONDARY KEY INDEXING ON TV-SERIES
PRESS 5 TO SEARCH TV-SERIES USING SECONDARY KEY
PRESS 6 TO INSERT A TV-SERIES TO DATASET
PRESS 7 TO DELETE A TV-SERIES FROM DATASET USING PRIMARY KEY
PRESS 8 TO DELETE A TV-SERIES FROM DATASET USING SECONDARY KEY

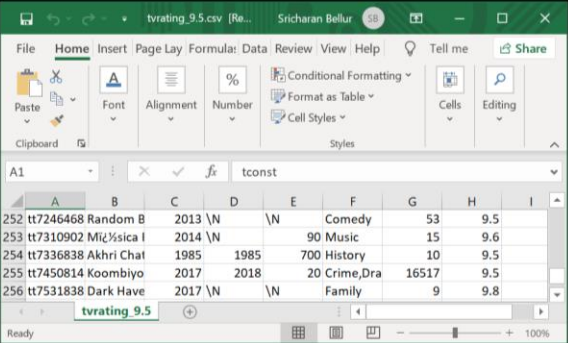
PRESS 0 TO EXIT
1
1
PRESS 1 to search a particular series by its title
PRESS 2 to Search the series according to start year
PRESS 3 to Search the series by its genre
PRESS 4 to Search by rating:
PRESS 0 to Exit
1
Enter the name of tv series you want to find :breaking bad
```

OBTAINING AND DISPLAYING MULTIPLE RESULT :

```
MINGW64/c/Users/sricharanvardhan/Desktop/fs mini project
PRESS 1 to search a particular series by its title
PRESS 2 to Search the series according to start year
PRESS 3 to Search the series by its genre
PRESS 4 to Search by rating:
PRESS 0 to Exit
1
Enter the name of tv series you want to find :breaking bad
Tconst : tt0903747
Tv series name : Breaking Bad
Start Year : 2008
End Year : 2013
Run time : 49
Genre: Crime,Drama,Thriller
Number of Votes : 151391
Rating : 9.4
Start Year : 2009
End Year : 2011
Run time : 5
Genre: Comedy,Crime,Drama
Number of Votes : 957
Rating : 7.8
\N indicates that the data has not been loaded
Tconst : tt4516518
Tv series name : Fixing Breaking Bad
Start Year : 2013
End Year : \N
Run time : \N
Genre: Comedy
Number of Votes : \N
Rating : \N
\N indicates that the data has not been loaded
Matches found :4
Completed the task in 0.99 seconds
```

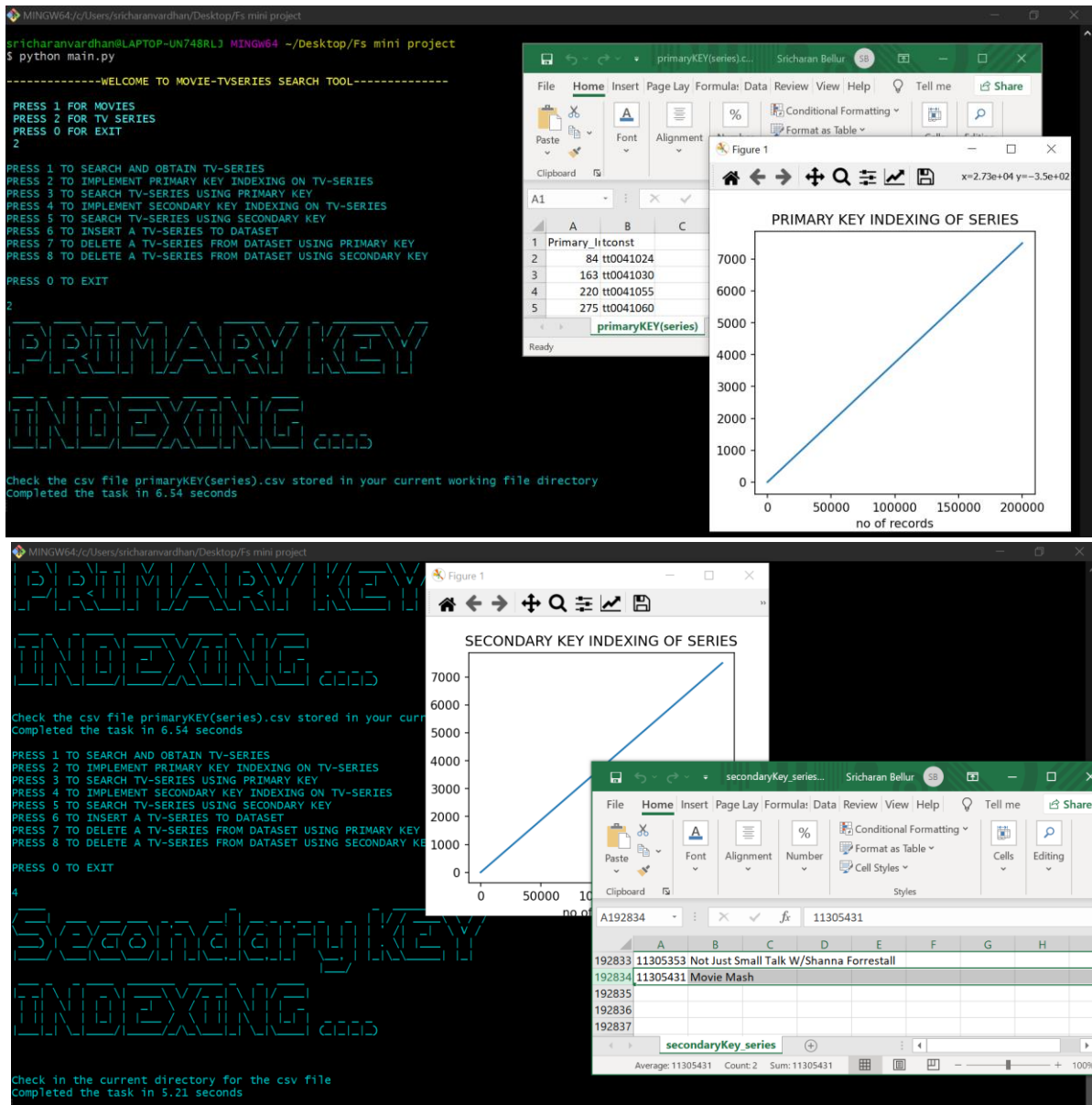
SEARCHING AND OBTAINING ALL THE SERIES BASED ON RATING:

```
Tconst : tt9686346
Tv series name : Chasing the Sun
Start Year : 2020
End Year : \N
Run time : \N
Genre: Documentary
Number of Votes : 151
Rating : 9.7
\N indicates that the data has not been loaded
Tconst : tt9716080
Tv series name : World's Greatest Festivals
Start Year : 2018
End Year : \N
Run time : \N
Genre: Adventure,Documentary
Number of Votes : 8
Rating : 9.6
\N indicates that the data has not been loaded
Matches found :299
Do you want to save it in a csv file? [1/0]
1
Check in the current directory for the csv file
```



| Rank | Title | Year | Genre | Votes | Rating |
|------|----------------------|------|-------|-------|--------|
| 252 | tt7246468 Random B | 2013 | \N | 53 | 9.5 |
| 253 | tt7310902 Miçusica l | 2014 | \N | 15 | 9.6 |
| 254 | tt7336838 Akhri Chat | 1985 | 1985 | 10 | 9.5 |
| 255 | tt7450814 Koombiyo | 2017 | 2018 | 16517 | 9.5 |
| 256 | tt7531838 Dark Have | 2017 | \N | 9 | 9.8 |

PRIMARY AND SECONDARY INDEXING(SERIES):



INSERTION AND DELETION OF SERIES :

The terminal window shows the execution of a Python script for inserting a TV series. The user enters the title 'jit series', start year '2018', end year '2022', runtime '480', genre 'Horror', number of votes '95', and average rating '10'. The script successfully inserts the series into the dataset.

The Excel spreadsheet, titled 'finalseries.csv', shows the resulting dataset. The series 'jit series' is added as a new row with the specified details.

| | A | B | C | D | E | F | G | H |
|--------|-----------|------------|------|------|-----|--------|----|----|
| 192936 | tt9916678 | Acelerado | 2019 | \N | | Comedy | \N | \N |
| 192937 | tt1111111 | jit series | 2018 | 2022 | 480 | Horror | 95 | 10 |
| 192938 | | | | | | | | |
| 192939 | | | | | | | | |
| 192940 | | | | | | | | |

The terminal window shows the execution of a Python script for deleting a TV series. The user enters the title 'jit series' and the start year '2018'. The script successfully deletes the series from the dataset.

The Excel spreadsheet, titled 'finalseries_deleted_secondary', shows the resulting dataset after the deletion. The series 'jit series' has been removed from the dataset.

| | A | B | C | D | E | F | G | H |
|--------|-----------|------------|------|----|----|--------------|-----|----|
| 192935 | tt9916380 | Meie aasti | 2019 | \N | | 43 Adventure | 104 | 9 |
| 192936 | tt9916678 | Acelerado | 2019 | \N | \N | Comedy | \N | \N |
| 192937 | | | | | | | | |
| 192938 | | | | | | | | |
| 192939 | | | | | | | | |
| 192940 | | | | | | | | |

CONCLUSIONS AND REFERENCES

CONCLUSIONS:

As stated above, the main objective have been implemented as effectively as possible in making this MOVIE-SEARCH TOOL a user friendly one. Taken ample of time to detect the errors and solve it to have a feasible mini project.

We have successfully implemented indexing which helps us in administrating the data used for managing the tasks performed. View tables are used to display all the components at once so that user can see all the components of a particular type at once. One can just select the component and modify and remove the component. We have successfully used various functionalities of python and created the File structures.

The terminal menu based project successfully carries out the above mentioned objectives hence concluding that the project is now feasible and can be accessed without a hassle.

The obtained results are compared into graphs, time and space constraints to check the efficiency and accessibility of this particular Movie/TV-Series finder tool.

REFERENCES:

- Stackoverflow(stackoverflow.com)
- GeeksforGeeks([GeeksforGeeks.com](https://www.geeksforgeeks.com))
- Tutorialspoint([tutorialspoint.com](https://www.tutorialspoint.com))
- Youtube([Youtube.com](https://www.youtube.com))
- Michael J.Folk, Bill Zoeclick, Greg Riccardi:File Structures-
An Object OrientedApproach with C++,3rd Edition,Pearson Education ,1998.
- Scott Robert Ladd:C++ Components and Algorithms,BPB Publications,1993