

Top Scorer: A Soccer Analytics Navigator

By

Kashish Shah (USC ID: 1842286535)

Saniya Kirkire (USC ID: 6354401648)

Shreyansh Khandelwal (USC ID: 6973534682)

FINAL REPORT

Introduction

Our goal was to create an Emulation Based File Distribution System (EDFS) with intuitive UI that will help user to perform all basic HDFS commands. User can upload a csv file onto the EDFS and partition it based on user argument. The user will be able to perform certain search and analytics functions that will use Partition based Map-Reduce to deduce the output.

Datasets:

The following three datasets will be used in this project, namely:

- **epl-goalScorer(20-21).csv**
- **LaLiga-goalScorer(20-21).csv**
- **Bundesliga-goalScorer(20-21).csv**

These datasets are part of ‘**Goal Dataset - Top 5 European Leagues**’ from Kaggle. They were scraped using python’s beautiful soup library.

Features of the Dataset include:

1. ID - unique ID for each row
2. player_name - name of the Soccer player
3. games - number of games played in the season
4. time - total number of minutes the player played that season
5. goals - number of goals scored in the season
6. xG - expected goals of the player
7. assists - number of assists provided in the season
8. xA - expected assists of the player
9. shots - number of shots taken during the season

EDFS Implementation:

EDFS has been emulated using two different ways. One of the ways involves the use of MySQL database while the other uses MongoDB.

A] EDFS implementation with SQL:

The following seven commands have been implemented in python scripts that will allow the user to make use of various functionalities like storing and retrieving data, creating, or deleting directories and viewing contents of the directories or file.

A namenode table has been created where the meta data associated with files or directories will be maintained based on the modifications made by the user to the file system. The data node maintains the actual data uploaded by the user.

The table has 4 columns:

- 1) Id – This field assigns an id to all the meta data. It is set as AUTO INCREMENT and PRIMARY KEY for the table.
- 2) Path – This field stores the path of all the Directories and Files created in the EDFS.
- 3) Name – This field stores the name of the file added to the EDFS.
- 4) Type – This field stores the type of the newly created file/directory.

id	path	name	fileType
44	root/table_eplss_1	table_eplss_1	csv
45	root/table_eplss_2	table_eplss_2	csv
46	root/table_eplss_3	table_eplss_3	csv
47	root/table_eeplass_1	table_eeplass_1	csv
48	root/table_eeplass_2	table_eeplass_2	csv
49	root/table_eeplass_3	table_eeplass_3	csv

6 rows in set (0.00 sec)

Fig. 1- Namenode

The actual data will be held in the data node. For example, if a user uploads a csv and decides to divide it in 3 partitions, we will create three datanodes with name ‘table_filename_1’, ‘table_filename_2’, and ‘table_filename_3’

Tables_in_dsci551
namenode3
new
new1
new2
new24
new3
new34
new4
table_dog_1
table_dog_2
table_dog_3
table_eplss_1
table_eplss_2
table_eplss_3
table_epl_1
table_epl_2
table_epl_3

Fig. 2 – csv being stored as Datanode

```

mysql> select * from table_eplss_1 limit 1;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | player_name | games | time | goals | xG | assists | xA | shots | key_passes | yellow_cards | red_cards | pos | team | npg | npXG | xGchain | xGBuildup |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 647 | Harry Kane | 35 | 3897 | 23 | 22.17485890910029 | 14 | 7.577093588188291 | 138 | 49 | 1 | 0 | F | Tottenham | 19 | 19.1301834397018 | 24.995648321695622 | 4.451256847940385 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> 

```

Fig. 3 – csv data as Datanode

EDFS commands:

- 1) **makeDir** – This function takes name as an argument from the user, and it does 2 tasks. Firstly, it creates a new folder or directory. Subsequently, it will add meta data created in the namenode.

```

~/Documents/551DB/Final Project > /opt/homebrew/bin/python3 "/Users/shreyanshkhanelwal/Documents/551DB/Final Project/hdfs.py"
Enter name of the directory: test
~/Documents/551DB/Final Project > 

```

id path	name	fileType
44 root/table_eplss_1	table_eplss_1	csv
45 root/table_eplss_2	table_eplss_2	csv
46 root/table_eplss_3	table_eplss_3	csv
47 root/table_eeplss_1	table_eeplss_1	csv
48 root/table_eeplss_2	table_eeplss_2	csv
49 root/table_eeplss_3	table_eeplss_3	csv
50 root/test	test	dir

Fig. 4 – Namenode after upload function

- 2) **ls** – This function simply just shows all the directories and files present. It takes no argument from user and outputs a list of paths to showcase the list of present files/directories.

```

120  ls()
121
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
~/Documents/551DB/Final Project > /opt/homebrew/bin/python3 "/Users/shreyanshkhanelwal/Documents/551DB/Final Project/hdfs.py"
('root/table_eplss_1',)
('root/table_eplss_2',)
('root/table_eplss_3')
('root/table_eeplss_1',)
('root/table_eeplss_2',)
('root/table_eeplss_3',)
('root/test')
~/Documents/551DB/Final Project > 

```

Fig. 5 ls function output

- 3) **cat** – This function takes name of the file as input from the user and outputs the content of the file. If the type of the file is a csv it will show a data frame in the output whereas if the type is txt, it will simply show all the content like a nano would.
For example, if the user gives input as ‘epl-goalScorer(20-21).csv’ as input it will show the data in a pandas data frame.

```

121     cat('eeplss')
122

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    JUPYTER

~/Documents/551DB/Final Project > /opt/homebrew/bin/python3 "/Users/shreyanshkhadelwal/Documents/551DB/Final Project/hdfs.py"
      Unnamed: 0   id   player_name  games  time  goals      xG ... red_cards  position  team_title  npg  npxG  xGChain  xGBuildup
0          0   647       Harry Kane  35  3097   23  22.174859 ...     0        F   Tottenham  19  19.130183  24.995648  4.451257
1          1  1250      Mohamed Salah  37  3085   22  20.250847 ...     0        F M S   Liverpool  16  15.683834  28.969234  9.800236
2          2  1228      Bruno Fernandes  37  3117   18  16.019454 ...     0        M S   Manchester United  9  8.407840  26.911412  11.932285
3          3  453       Son Heung-Min  37  3139   17  11.023287 ...     0        F M S   Tottenham  16  10.262118  20.671916  6.608751
4          4  822       Patrick Bamford  38  3085   17  18.401863 ...     0        F S      Leeds  15  16.879525  23.394953  4.131796
..        ...
517        517  9415     Jaden Philogene-Bidace  1   1   0  0.000000 ...     0        S   Aston Villa  0  0.000000  0.056044  0.056044
518        518  9423      Gaetano Berardi  2   113   0  0.074761 ...     0        D S      Leeds  0  0.074761  0.231278  0.231278
519        519  9524      Anthony Elanga  1   67   0  0.000000 ...     0        M   Manchester United  0  0.000000  0.000000  0.000000
520        520  9540       Femi Seriki  1   1   0  0.000000 ...     0        S   Sheffield United  0  0.000000  0.000000  0.000000
521        521  9552      Tyrese Francois  1   13   0  0.000000 ...     0        S      Fulham  0  0.000000  0.000000  0.000000
[522 rows x 19 columns]
~/Documents/551DB/Final Project > []

```

Fig 6. Concat output

- 4) **remove** – This function simply just deletes the directory or file specified by the user. It takes name of the file as an argument from user and deletes it from both the namenode and datanode table.

```

mysql> select * from namenode3;
+----+-----+-----+-----+
| id | path           | name        | fileType |
+----+-----+-----+-----+
| 47 | root/table_eeplss_1 | table_eeplss_1 | csv      |
| 48 | root/table_eeplss_2 | table_eeplss_2 | csv      |
| 49 | root/table_eeplss_3 | table_eeplss_3 | csv      |
| 50 | root/test         | test         | dir      |
+----+-----+-----+-----+
4 rows in set (0.01 sec)

```

Fig 7. Namenode output before delete

```

127     remove()
128

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    JUPYTER

~/Documents/551DB/Final Project > /opt/homebrew/bin/python3 "/Users/shreyanshkhadelwal/Documents/551DB/Final Project/hdfs.py"
Enter name of the directory or file you want to delete: eeplss
removed
~/Documents/551DB/Final Project > []

```

```

mysql> select * from namenode3;
+----+-----+-----+-----+
| id | path           | name        | fileType |
+----+-----+-----+-----+
| 50 | root/test         | test         | dir      |
+----+-----+-----+-----+
1 row in set (0.00 sec)

```

Fig 8. Namenode output after delete

- 5) **Upload** – This function will ask the user to upload the csv files. It will then update 2 tables. Firstly, it will create an entry in the namenode and then it will store the data in the datanode table.

```
[mysql> select * from namenode3;
+----+-----+-----+-----+
| id | path           | name      | fileType |
+----+-----+-----+-----+
| 50 | root/test      | test      | dir      |
| 51 | root/table_eplGoal_1 | table_eplGoal_1 | csv      |
| 52 | root/table_eplGoal_2 | table_eplGoal_2 | csv      |
| 53 | root/table_eplGoal_3 | table_eplGoal_3 | csv      |
+----+-----+-----+-----+
4 rows in set (0.00 sec)
```

Fig 9. Name node data after upload operation

```
[mysql> show tables;
+-----+
| Tables_in_dsci551 |
+-----+
| namenode3          |
| table_eplGoal_1    |
| table_eplGoal_2    |
| table_eplGoal_3    |
+-----+
4 rows in set (0.01 sec)
```

Fig 10. Data gets stored in parts

```
mysql> select * from table_eplGoal_3 limit 5;
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | player_name | games | time | goals | xG | assists | xA | shots | key_passes | yellow_cards | red_cards | pos | team | npxG | npg | npXG | xGChain | xGBuildup |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1699 | Shkodran Mustafi | 3 | 43 | 0 | 0.0 | 0 | 0.0 | 0 | 0 | 0 | 0 | 0 | Arsenal | 0 | 0.0 | 0.0647247657179832 | 0.0647247657179832 |
| 1700 | Javier Manquillo | 13 | 823 | 0 | 0.0231618583202362 | 0 | 0.2976967543363571 | 1 | 5 | 1 | 0 | Newcastle United | 0 | 0.0231618583202362 | 1.4320279778912663 | 1.11116935295557 |
| 1735 | Jack Stephens | 18 | 1544 | 0 | 0.8943833330162498 | 0 | 0.455656368348369 | 0 | 6 | 2 | 0 | Southampton | 0 | 0.8943833330162498 | 3.081087851896882 | 2.751403872848916 |
| 1739 | Eric Bailly | 11 | 826 | 0 | 0.0487859137356281 | 0 | 0.0 | 1 | 0 | 3 | 0 | Manchester United | 0 | 0.0487859137356281 | 1.188666683171272 | 1.1886668683171272 |
| 1747 | Kevin Long | 8 | 635 | 0 | 0.6100033670663834 | 0 | 0.0173680447041988 | 2 | 1 | 2 | 0 | Burnley | 0 | 0.6100033670663834 | 0.5022359008061266 | 0.5022359008061266 |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)
```

Fig 11. Data in one of the partitions

- 6) **getPartitionLoc** – This function will allow users to get path of their file partitions that is created by the EDFS.

```
128   getPartitionLoc('eplGoal',3)

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER
~/Documents/551DB/Final Project > /opt/homebrew/bin/python3 "/Users/shreyanshkhanelwal/Documents/551DB/Final Project/hdfs.py"
root/table_eplGoal_3
~/Documents/551DB/Final Project >
```

Fig 12. Path of the partition is displayed as an output

- 7) **readPartition** – This function will read a particular partition of the uploaded csv and return the data frame of that partition.

Fig 12. Reading individual partition

B] EDFS implementation with MongoDB:

In MongoDB, the namenode with meta data and the data node is maintained in the form of collections. The namenode collection will include the path, file type and name of the file uploaded or directory created by the user.

- 1) `makedir` - the name of the directory to be created will be passed as an input to the function. The namenode collection will be updated with the metadata of the directory that has been created.

```

dsci551> db.namenode.find({})
[
  {
    "_id": ObjectId("637b60b2f68b0f8c7064192d"),
    "path": "root/table_epl_1",
    "name": "table_epl_1",
    "fileType": "csv"
  },
  {
    "_id": ObjectId("637d6c9fd72912fc1d6e1449"),
    "path": "root/table_eplGoal_1",
    "name": "table_eplGoal_1",
    "fileType": "csv"
  },
  {
    "_id": ObjectId("637d6c9fd72912fc1d6e144a"),
    "path": "root/table_eplGoal_2",
    "name": "table_eplGoal_2",
    "fileType": "csv"
  },
  {
    "_id": ObjectId("637d6c9fd72912fc1d6e144b"),
    "path": "root/table_eplGoal_3",
    "name": "table_eplGoal_3",
    "fileType": "csv"
  },
  {
    "_id": ObjectId("637f1a03678ee7214fb304f"),
    "path": "root/table_eplGoal_1",
    "name": "table_eplGoal_1",
    "fileType": "csv"
  },
  {
    "_id": ObjectId("637f1a03678ee7214fb3050"),
    "path": "root/table_eplGoal_2",
    "name": "table_eplGoal_2",
    "fileType": "csv"
  },
  {
    "_id": ObjectId("637f1a03678ee7214fb3051"),
    "path": "root/table_eplGoal_3",
    "name": "table_eplGoal_3",
    "fileType": "csv"
  },
  {
    "_id": ObjectId("6383f048dc16a6638e2a4b5"),
    "path": "root/Goal",
    "name": "Goal",
    "fileType": "dir"
  }
]
dsci551> []

```

Fig 13. Name node after creating files

- 2) ls - The file contents inside the directory will be listed. No input is required in order to execute this function.

```

117  ls()
118  |

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER
~/Documents/551DB/Final Project main !2 ?14 > /opt/homebrew/bin/python3 "/Users/shreyanshkhadelwal/Documents/551DB/Final Project/mongo.py"
Success
{'path': 'root/table_epl_1'}
{'path': 'root/table_eplGoal_1'}
{'path': 'root/table_eplGoal_2'}
{'path': 'root/table_eplGoal_3'}
{'path': 'root/table_eplGoal_1'}
{'path': 'root/table_eplGoal_2'}
{'path': 'root/table_eplGoal_3'}
{'path': 'root/Goal'}
~/Documents/551DB/Final Project main !2 ?14 > []

```

Fig 14. Displaying contents of directory

- 3) cat - This function will be used to simply display the contents of a file that has been input by the user.

```

117 cat('eplGoals')
118 |
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
~/Documents/551DB/Final Project main !2 ?14 > /opt/homebrew/bin/python3 "/Users/shreyanshkhadelwal/Documents/551DB/Final Project/mongo.py"
Success
      Unnamed: 0   id     player_name  games  time  goals    xG ... red_cards  position      team_title  npg    npxG  xGChain  xGBuildup
0       0   647      Harry Kane    35 3097    23 22.174859 ...      0        F  Tottenham  19 19.130183 24.995648  4.451257
1       1  1250     Mohamed Salah   37 3085    22 20.250847 ...      0        F M S  Liverpool  16 15.683834 28.968234  9.800236
2       2  1228     Bruno Fernandes  37 3117    18 16.019454 ...      0        M S  Manchester United  9 8.407840 26.911412 11.932285
3       3   453      Son Heung-Min  37 3139    17 11.023287 ...      0        F M S  Tottenham  16 10.262118 20.671916  6.608751
4       4   822      Patrick Bamford  38 3085    17 18.401863 ...      0        F S  Leeds  15 16.879525 23.394953  4.131796
...   ...
517    517  9415    Jaden Philogene-Bidace  1   1    0  0.000000 ...      0        S  Aston Villa  0  0.000000  0.056044  0.056044
518    518  9423    Gaetano Berardi   2  113    0  0.074761 ...      0        D S  Leeds  0  0.074761  0.231278  0.231278
519    519  9524    Anthony Elanga   1   67    0  0.000000 ...      0        M  Manchester United  0  0.000000  0.000000  0.000000
520    520  9540    Femi Seriki   1   1    0  0.000000 ...      0        S  Sheffield United  0  0.000000  0.000000  0.000000

```

Fig 15. Displaying contents using cat operation

- 4) remove – The name of the file to be deleted will be passed as an input by the user. the function will simply delete the collection that holds the data associated with the file name and the metadata in the namenode collection will be updated accordingly.

```

117 remove()
118 |
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
~/Documents/551DB/Final Project main !2 ?14 > /opt/homebrew/bin/python3 "/Users/shreyanshkhadelwal/Documents/551DB/Final Project/mongo.py"
Success
Enter name of the directory or file you want to delete: project
removed
~/Documents/551DB/Final Project main !2 ?14 >

```

Fig 16. Deleting a folder or directory

- 5) upload - the name of the file and the number of partitions will be passed as an input by the user. the file will be stored in parts in different collection depending upon the number of partitions input. The metadata will be maintained in the name node collection.

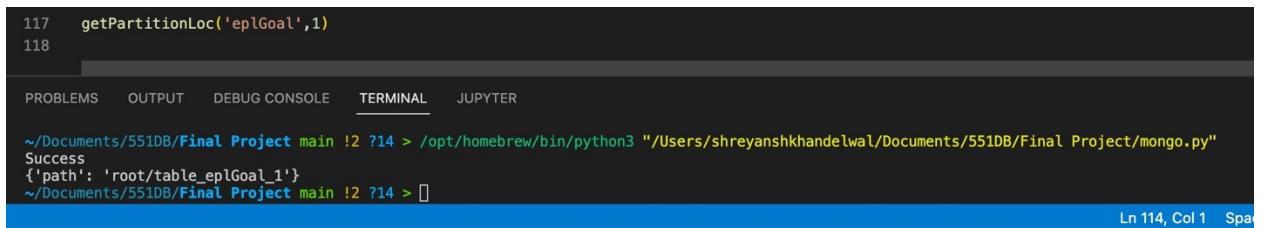
```

]
dsci551> db.table_eplGoal_1.find({})
[ {
  '_id': {
    '0': {
      id: 647,
      player_name: 'Harry Kane',
      games: 35,
      time: 3097,
      goals: 23,
      xG: 22.1748589091,
      assists: 5,
      xA: 19.7035882,
      shots: 138,
      key_passes: 49,
      yellow_cards: 1,
      red_cards: 0,
      position: 'F',
      team_title: 'Tottenham',
      nppg: 19,
      npxG: 19.1301834397,
      xGChain: 24.995648327,
      xGBuildup: 4.4512568479
    },
    '1': {
      id: 1259,
      player_name: 'Mohamed Salah',
      games: 37,
      time: 3085,
      goals: 22,
      xG: 20.2508466374,
      assists: 5,
      xA: 6.5285261013,
      shots: 126,
      key_passes: 55,
      yellow_cards: 0,
      red_cards: 0,
      position: 'F M S',
      team_title: 'Liverpool',
      nppg: 16,
      npxG: 15.6838336382,
      xGChain: 28.9682335481,
      xGBuildup: 9.8002356552
    },
    '2': {
      id: 1228,
      player_name: 'Bruno Fernandes',
      games: 37,
      time: 3117,
      goals: 18,
      xG: 16.01945464615,
      assists: 12,
      xA: 11.4749959782,
      shots: 121,
      key_passes: 95,
      yellow_cards: 6,
      red_cards: 0,
      position: 'M S',
      team_title: 'Manchester United',
      nppg: 9,
      npxG: 8.407839627,
      xGChain: 26.9114123788,
      xGBuildup: 11.9322848711
    },
    '3': {
  }
}

```

Fig 17. Datanode after upload

- 6) `getPartitionLoc` - The location of the file name that has been passed as an input by the user will be returned. The path can be returned by querying the namenode collection.



```

117     getPartitionLoc('eplGoal',1)
118
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   JUPYTER
~/Documents/551DB/Final Project main !2 ?14 > /opt/homebrew/bin/python3 "/Users/shreyanshkhanelwal/Documents/551DB/Final Project/mongo.py"
Success
{'path': '/root/table_eplGoal_1'}
~/Documents/551DB/Final Project main !2 ?14 > []

```

Fig 18. Path of the partition of the file

- 7) `readPartition` - The contents of a specific partition of the file name input by the user will be displayed. The file name and the part to be displayed will be input by the user.

```

117   readPartition['eplGoal',1]
118
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    JUPYTER
~Documents/551DB/Final Project main i2 ?14 > /opt/homebrew/bin/python3 "/Users/shreyanshkhadelwal/Downloads/551DB/Final Project/mongo.py"
Success
{
  "_id": ObjectId('637f1a03678ee7214fb3052'),
  "0": {"id": 647, "player_name": "Harry Kane", "games": 35, "time": 3097, "goals": 23, "xG": 22.1748589091, "assists": 14, "xA": 7.5770935882, "shots": 138, "key_passes": 49, "yellow_cards": 1, "red_cards": 0, "position": "F", "team_title": "Tottenham", "npxG": 19, "npxG": 19.1301834397, "xGChain": 24.0956483217, "xGBuildup": 4.4512568479}, "1": {"id": 1250, "player_name": "Mohamed Salah", "games": 37, "time": 3085, "goals": 22, "xG": 20.2508466374, "assists": 5, "xA": 6.5285261013, "shots": 126, "key_passes": 55, "yellow_cards": 0, "red_cards": 0, "position": "F M S", "team_title": "Liverpool", "npxG": 16, "npxG": 15.6838336382, "xGChain": 28.9662335481, "xGBuildup": 9.8002356552}, "2": {"id": 1228, "player_name": "Bruno Fernandes", "games": 37, "time": 3117, "goals": 18, "xG": 16.0194544615, "assists": 12, "xA": 11.4749950782, "shots": 121, "key_passes": 95, "yellow_cards": 6, "red_cards": 0, "position": "M S", "team_title": "Manchester United", "npxG": 9, "npxG": 8.407839627, "xGChain": 26.91114123788, "xGBuildup": 11.9322848711}, "3": {"id": 453, "player_name": "Son Heung-Min", "games": 37, "time": 3139, "goals": 17, "xG": 11.0232867561, "assists": 10, "xA": 9.5129924696, "shots": 68, "key_passes": 75, "yellow_cards": 6, "red_cards": 0, "position": "F M S", "team_title": "Tottenham", "npxG": 16, "npxG": 10.2621179037, "xGChain": 20.6719155498, "xGBuildup": 6.6087511387}, "4": {"id": 822, "player_name": "Patrick Bamford", "games": 38, "time": 3085, "goals": 17, "xG": 18.4018626101, "assists": 7, "xA": 7, "shots": 107, "key_passes": 30, "yellow_cards": 3, "red_cards": 0, "position": "F S", "team_title": "Leeds", "npxG": 15, "npxG": 15.8795249462, "xGChain": 23.394952707, "xGBuildup": 4.131795546}, "5": {"id": 5555, "player_name": "Dominic Calvert-Lewin", "games": 32, "time": 2788, "goals": 16, "xG": 18.1364487149, "assists": 16, "xA": 11.4749950782, "shots": 82, "key_passes": 18, "yellow_cards": 3, "red_cards": 0, "position": "EVERTON", "npxG": 16, "npxG": 16.0194544615, "xGChain": 19.6988500208, "xGBuildup": 4.159406215}, "6": {"id": 755, "player_name": "Jamie Vardy", "games": 34, "time": 2848, "goals": 15, "xG": 19.9429461099, "assists": 9, "xA": 5.0878815223, "shots": 82, "key_passes": 28, "yellow_cards": 1, "red_cards": 0, "position": "F S", "team_title": "Leicester", "npxG": 7, "npxG": 13.0924265794, "xGChain": 18.2279073671, "xGBuildup": 2.4125877339}, "7": {"id": 8865, "player_name": "Ollie Watkins", "games": 37, "time": 3330, "goals": 14, "xG": 16.2801776454, "assists": 5, "xA": 5.3140294915, "shots": 98, "key_passes": 107, "yellow_cards": 2, "red_cards": 0, "position": "F", "team_title": "Aston Villa", "npxG": 13, "npxG": 14.757840015, "xGChain": 22.0140207857, "xGBuildup": 5.3349911738}, "8": {"id": 314, "player_name": "Ilkay Gündogan", "games": 28, "time": 2033, "goals": 13, "xG": 9.5668838086, "assists": 2, "xA": 3.8162196558, "shots": 54, "key_passes": 42, "yellow_cards": 1, "red_cards": 0, "position": "M S", "team_title": "Manchester City", "npxG": 12, "npxG": 8.4454543459, "xGChain": 19.2868117318, "xGBuildup": 10.1483747363}, "9": {"id": 3277, "player_name": "Alexandre Lacazette", "games": 30, "time": 1928, "goals": 13, "xG": 12.0289094187, "assists": 2, "xA": 2.1018188335, "shots": 45, "key_passes": 22, "yellow_cards": 3, "red_cards": 0, "position": "F M S", "team_title": "Arsenal", "npxG": 10, "npxG": 9.7454028763, "xGChain": 14.3615423944, "xGBuildup": 2.9379407838}, "10": {"id": 468, "player_name": "Callum Wilson", "games": 26, "time": 2079, "goals": 12, "xG": 13.5900912508, "assists": 5, "xA": 3.046300631, "shots": 49, "key_passes": 28, "yellow_cards": 1, "red_cards": 0, "position": "F S", "team_title": "Newcastle United", "npxG": 8, "npxG": 10.54541587, "xGChain": 14.2055929303, "xGBuildup": 1.0525451973}, "11": {"id": 620, "player_name": "Kelechi Iheanacho", "games": 25, "time": 1452, "goals": 12, "xG": 9.0485961484, "assists": 2, "xA": 3.5340991933, "shots": 59, "key_passes": 59, "yellow_cards": 1, "red_cards": 0, "position": "F S", "team_title": "Leicester", "npxG": 12, "npxG": 8.28742723, "xGChain": 13.9615636356, "xGBuildup": 3.4299153809}, "12": {"id": 986, "player_name": "Danny Ings", "games": 28, "time": 2115, "goals": 12, "xG": 8.1392292418, "assists": 4, "xA": 2.3112305803, "shots": 56, "key_passes": 22, "yellow_cards": 1, "red_cards": 0, "position": "F S", "team_title": "Southampton", "npxG": 10, "npxG": 6.6168915182, "xGChain": 9.9290986912, "xGBuildup": 2.0632555448}, "13": {"id": 4456, "player_name": "Chris Wood", "games": 32, "time": 2675, "goals": 12, "xG": 12.8960091174, "assists": 3, "xA": 2.0381183885, "shots": 69, "key_passes": 23, "yellow_cards": 0, "red_cards": 0, "position": "F S", "team_title": "Burnley", "npxG": 10, "npxG": 11.3736713827, "xGChain": 13.5852815248, "xGBuildup": 2.1498517599}, "14": {"id": 522, "player_name": "Wilfried Zaha", "games": 30, "time": 2615, "goals": 11, "xG": 8.8044398949, "assists": 2, "xA": 3.7028211039, "shots": 60, "key_passes": 30, "yellow_cards": 6, "red_cards": 0, "position": "F M S", "team_title": "Crystal Palace", "npxG": 9, "npxG": 7.2821022347, "xGChain": 13.5794724748, "xGBuildup": 3.6344361883}, "15": {"id": 556, "player_name": "Marcus Rashford", "games": 37, "time": 2941, "goals": 11, "xG": 9.5797099918, "assists": 9, "xA": 4.1851221155, "shots": 79, "key_passes": 44, "yellow_cards": 4, "red_cards": 0, "position": "F M S", "team_title": "Manchester United", "npxG": 11, "npxG": 9.5797099918, "xGChain": 20.4419101626, "xGBuildup": 8.9272075575}, "16": {"id": 838, "player_name": "Sadio Mané", "games": 35, "time": 2885, "goals": 11, "xG": 14.8285477553, "assists": 7, "xA": 7.7877548542, "shots": 94, "key_passes": 61, "yellow_cards": 3, "red_cards": 0, "position": "F M S", "team_title": "Liverpool", "npxG": 11, "npxG": 14.8285477553, "xGChain": 24.9989231974, "xGBuildup": 6.0576562826}, "17": {"id": 2251, "player_name": "Gareth Bale", "games": 20, "time": 909, "goals": 11, "xG": 5.7978576776, "assists": 2, "xA": 1.8422413953, "shots": 38, "key_passes": 21, "yellow_cards": 1, "red_cards": 0, "position": "M S", "team_title": "Tottenham", "npxG": 11, "npxG": 5.7978576776, "xGChain": 8.1496490103, "xGBuildup": 2.594513725}, "18": {"id": 7153, "player_name": "Matheus Pereira", "games": 33, "time": 2594, "goals": 11, "xG": 6.9483165387, "assists": 19 rows in set (0.01 sec)

```

Fig 19. reading content of a particular partition of a file

PMR

Three search analytics functions have been implemented using the concept of map reduce. The search analytics functions will be executed on every individual partition of a file. The results obtained will be combined and the final response to the query will be returned to the user.

- Function 1:** Search and return a list of names of players who have scored goals greater than user input

```

mysql> select player_name from table_eeplass_1 where goals>10;
+-----+
| player_name |
+-----+
| Harry Kane |
| Mohamed Salah |
| Bruno Fernandes |
| Son Heung-Min |
| Patrick Bamford |
| Dominic Calvert-Lewin |
| Jamie Vardy |
| Ollie Watkins |
| Ilkay Gündogan |
| Alexandre Lacazette |
| Callum Wilson |
| Kelechi Iheanacho |
| Danny Ings |
| Chris Wood |
| Wilfried Zaha |
| Marcus Rashford |
| Sadio Mané |
| Gareth Bale |
| Matheus Pereira |
+-----+
19 rows in set (0.01 sec)

```

```
17 goals('eeplass', 1, 10)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

~/Documents/551DB/Final Project > /opt/homebrew/bin/python3 "/Users/shreyanshkhanelwal/Documents/551DB/Final Project/pmr.py"
[('Harry Kane',), ('Mohamed Salah',), ('Bruno Fernandes',), ('Son Heung-Min',), ('Patrick Bamford',), ('Dominic Calvert-Lewin',), ('Jamie Vardy',), ('Ollie Watkins',), ('Ilkay Gündogan',), ('Alexandre Lacazette',), ('Callum Wilson',), ('Kelechi Iheanacho',), ('Danny Ings',), ('Chris Wood',), ('Wilfried Zaha',), ('Marcus Rashford',), ('Sadio Mané',), ('Gareth Bale',), ('Matheus Pereira',)]
~/Documents/551DB/Final Project >
```

Fig 20. Output of first search and analytics function

- **Function 2:** Search and return a list of names of players who have collected xG between the range provided by the user

```
mysql> select player_name, goals from table_eeplass_1 where xG>10 and xG<15;
+-----+-----+
| player_name | goals |
+-----+-----+
| Son Heung-Min | 17 |
| Alexandre Lacazette | 13 |
| Callum Wilson | 12 |
| Chris Wood | 12 |
| Sadio Mané | 11 |
| Pierre-Emerick Aubameyang | 10 |
| Michail Antonio | 10 |
| Raheem Sterling | 10 |
| Roberto Firmino | 9 |
| Che Adams | 9 |
| Neal Maupay | 8 |
| Richarlison | 7 |
| Timo Werner | 6 |
+-----+-----+
13 rows in set (0.00 sec)

mysql>
```

```
24 xG('eeplass', 1, 10, 15)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

~/Documents/551DB/Final Project > /opt/homebrew/bin/python3 "/Users/shreyanshkhanelwal/Documents/551DB/Final Project/pmr.py"
[('Son Heung-Min',), ('Alexandre Lacazette',), ('Callum Wilson',), ('Chris Wood',), ('Sadio Mané',), ('Pierre-Emerick Aubameyang',), ('Michail Antonio',), ('Raheem Sterling',), ('Roberto Firmino',), ('Che Adams',), ('Neal Maupay',), ('Richarlison',), ('Timo Werner',)]
~/Documents/551DB/Final Project >
```

Fig 21. Output of second search and analytics function

- **Function 3:** Return a list of players who have scored the number of goals given by the user.

```
mysql> select player_name, goals from table_eeplass_1 where goals=13;
+-----+
| goals |
+-----+
| Ilkay Gündogan |
| Alexandre Lacazette |
+-----+
2 rows in set (0.00 sec)

mysql>
```

```

32 searchGoals('eep1ss', 1, 13)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

~/Documents/551DB/Final Project > /opt/homebrew/bin/python3 "/Users/shreyanshkhanelwal/Documents/551DB/Final Project/pmr.py"
[('Ilkay Gündogan',), ('Alexandre Lacazette',)]
~/Documents/551DB/Final Project > [ ]

```

Fig 22. Output of third search and analytics function

Web Interface for user:

A user-friendly interface has been implemented. The left page has a menu which offers the user to select from the seven different it's HDFS functionalities. Based on the function selected from the menu, the user will be directed to a web page. The user will be prompted to pass the required input parameters. Technologies used are HTML CSS JavaScript and Python flask. Flask server act as a bridge between the front end and the back end. The Python script will be executed in the back end based on the user input.

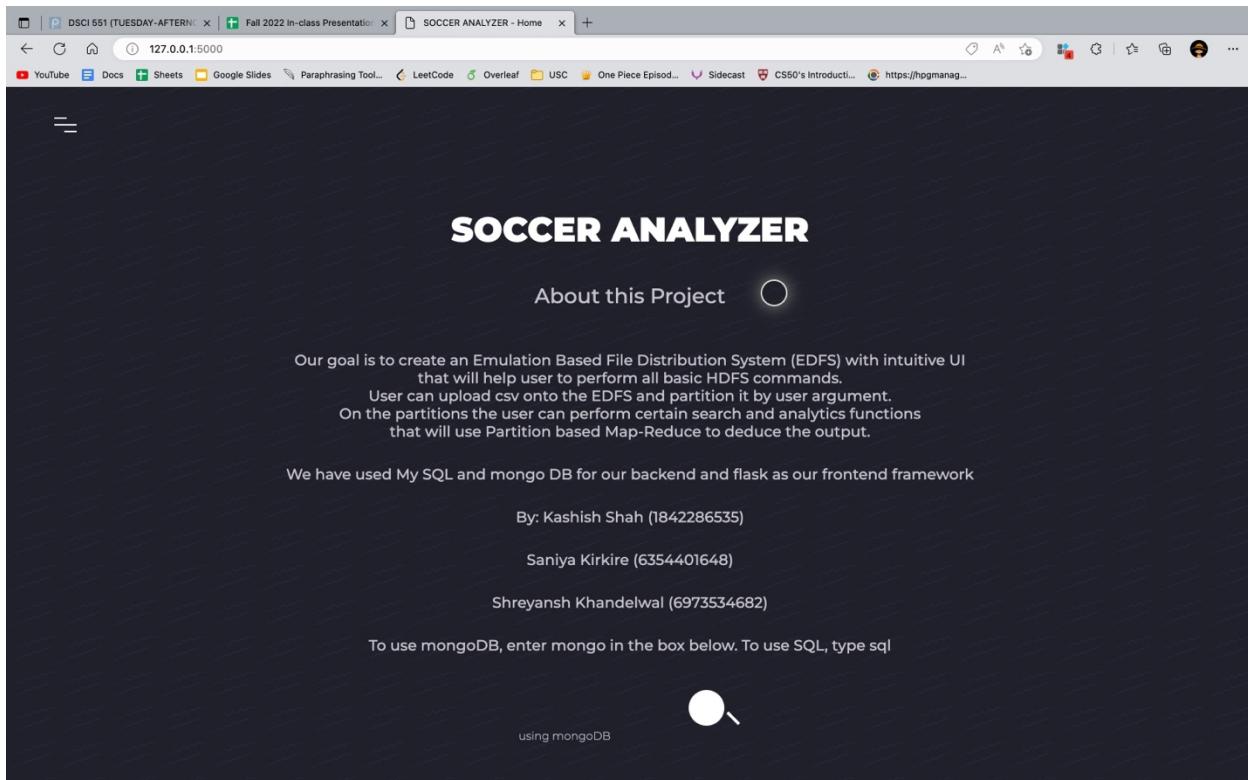


Fig 23. Website Homepage

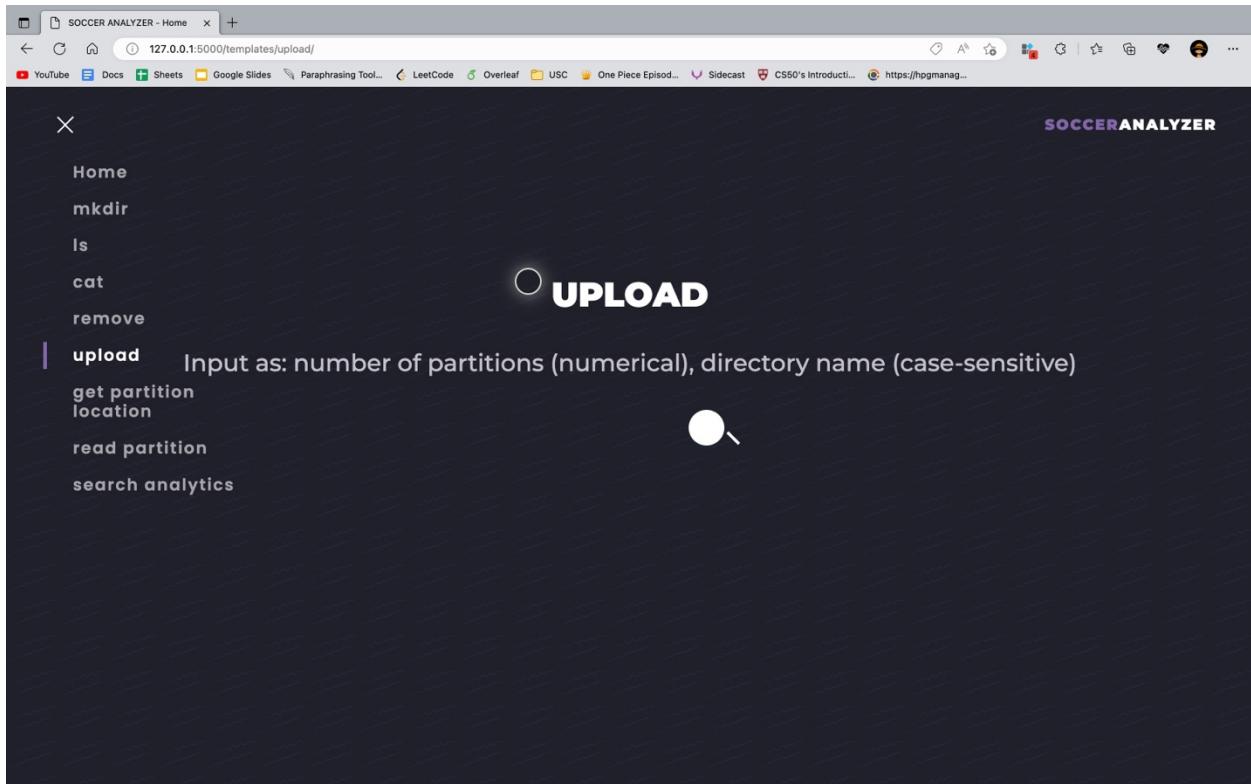


Fig 24. Upload function

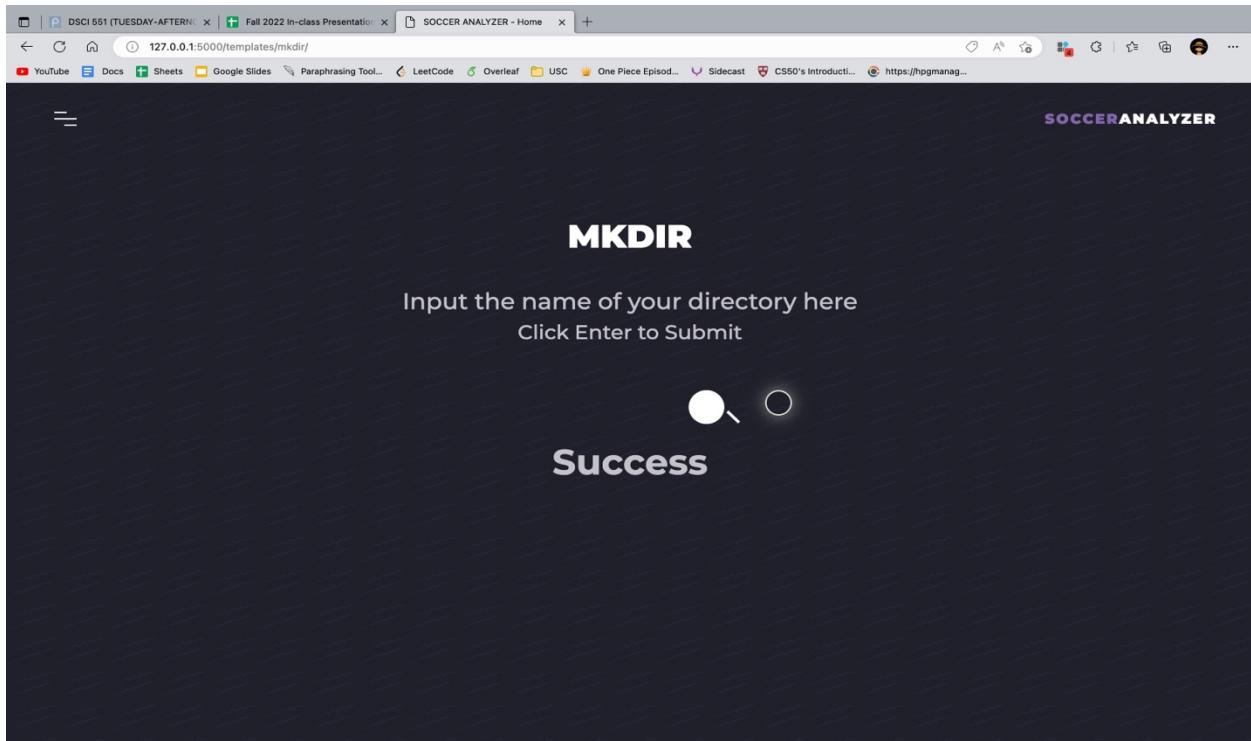


Fig 25. Make directory command

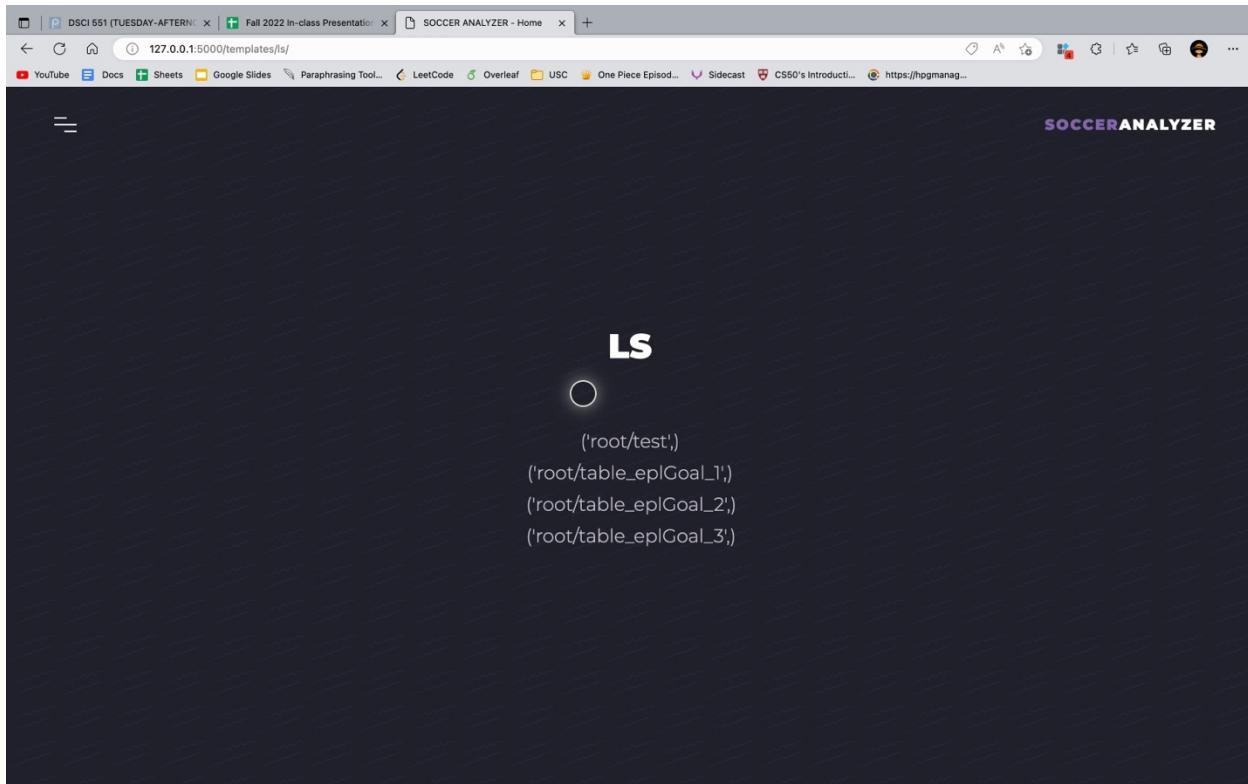


Fig 26. Ls command

ID	player_name	games	time	goals	xG	assists	xA	shots	key_passes	yellow_cards	red_cards	pos	team	mpg	npxc	xGchain	xGbuildup
0	Harry Kane	35	3097	23	22.17485890910029	14	7.577093588188291	138	49	1	0	F	Tottenham	19	19.1301834397018	24.995648321695622	4.4512568
1	Mohamed Salah	37	3085	22	20.250846637412906	5	6.528526101261377	126	55	0	0	F M S	Liverpool	16	15.683833638206124	28.968233548107463	9.8002350
2	Bruno Fernandes	37	3117	18	16.0194544615224	12	11.474995978176594	121	95	6	0	M S	Manchester United	9	8.40783962700516	26.911412378773093	11.9322848
3	Son Heung-Min	37	3139	17	11.023286756128073	10	9.5129924695933884	68	75	0	0	F M S	Tottenham	16	10.26211790367961	20.6719155497849	6.6087513
4	Patrick Bamford	38	3085	17	18.401862610131506	7	3.7822465524077415	107	30	3	0	F S	Leeds	15	16.87952494621277	23.394952706992623	4.13719554
5	Dominic Calvert-Lewin	32	2788	16	18.136448714882132	0	2.101818833500147	82	18	3	0	F S	Everton	16	18.136448714882132	19.69885002076626	4.1594062
6	Jamie Vardy	34	2848	15	19.94294610992074	9	5.08788152253156	82	28	1	0	F S	Leicester	7	13.0924265794456	18.227907367050648	2.41258775
7	Ollie Watkins	37	3330	14	16.280177645385265	5	5.314029491506517	98	45	2	1	F	Aston Villa	13	14.757840014994144	22.014020785689357	5.33499117
8	Illkay Gündogan	28	2033	13	9.566883080638943	2	3.8162196557968855	54	42	1	0	M S	Manchester City	12	8.044543545939696	19.286811731755733	10.1483747
9	Alexandre Lacazette	30	1928	13	12.028909418731928	2	2.209213115274906	45	22	3	0	F M S	Arsenal	10	9.7454028762877	14.361542394384742	2.9379407
10	Callum Wilson	26	2079	12	13.59009125083685	5	3.046300631016493	49	17	2	0	F S	Newcastle United	8	10.545415870843518	14.205592930316923	1.05254519
11	Kelechi Iheanacho	25	1452	12	9.048596148379149	2	3.534099913289876	59	28	1	0	F S	Leicester	12	8.287427507031845	13.961563635671496	3.4299155
12	Danny Ings	28	2115	12	8.139229241758585	4	2.3112305803224444	56	22	1	0	F S	Southampton	10	6.616891518235207	9.92908869124949	2.0632555
13	Chris Wood	32	2675	12	13.896009117364833	3	2.0381838850379	69	23	0	0	F S	Burnley	10	11.373671362666534	13.585281524807217	2.14985175
14	Wilfried Zaha	30	2615	11	8.804439894855022	2	3.702821103855968	60	30	6	0	F M S	Crystal Palace	9	7.282102234661579	13.579472474753857	3.63443618
15	Marcus Rashford	37	2941	11	9.579709991812706	9	4.185122115537524	79	44	4	0	F M S	Manchester United	11	9.579709991812706	20.44191016256809	8.9272075
16	Sadio Mané	35	2805	11	14.282547755256295	7	7.787754854187369	94	61	3	0	F M S	Liverpool	11	14.282547755256295	24.99892319738865	6.0576562
17	Gareth Bale	20	909	11	5.797857677564025	2	1.842241395264864	38	21	1	0	M S	Tottenham	11	5.797857677564025	8.149640910327435	2.59451372
18	Matheus Pereira	33	2594	11	6.948316538706422	6	6.355539299547672	65	58	2	1	M S	West Bromwich Albion	7	3.9036410860717297	10.857054157182574	3.75792377
19	Pierre-Emerick Aubameyang	28	2254	10	10.194088630378246	3	2.3524065418168902	55	19	2	0	F M S	Arsenal	8	8.677150970184803	14.778997933492064	3.95512931
20	Michail Antonio	25	1923	10	14.138838540762665	5	3.034944863989949	63	25	3	0	F S	West Ham	10	14.138838540762665	18.9988911146331	2.6550637

Fig 27. Cat command output

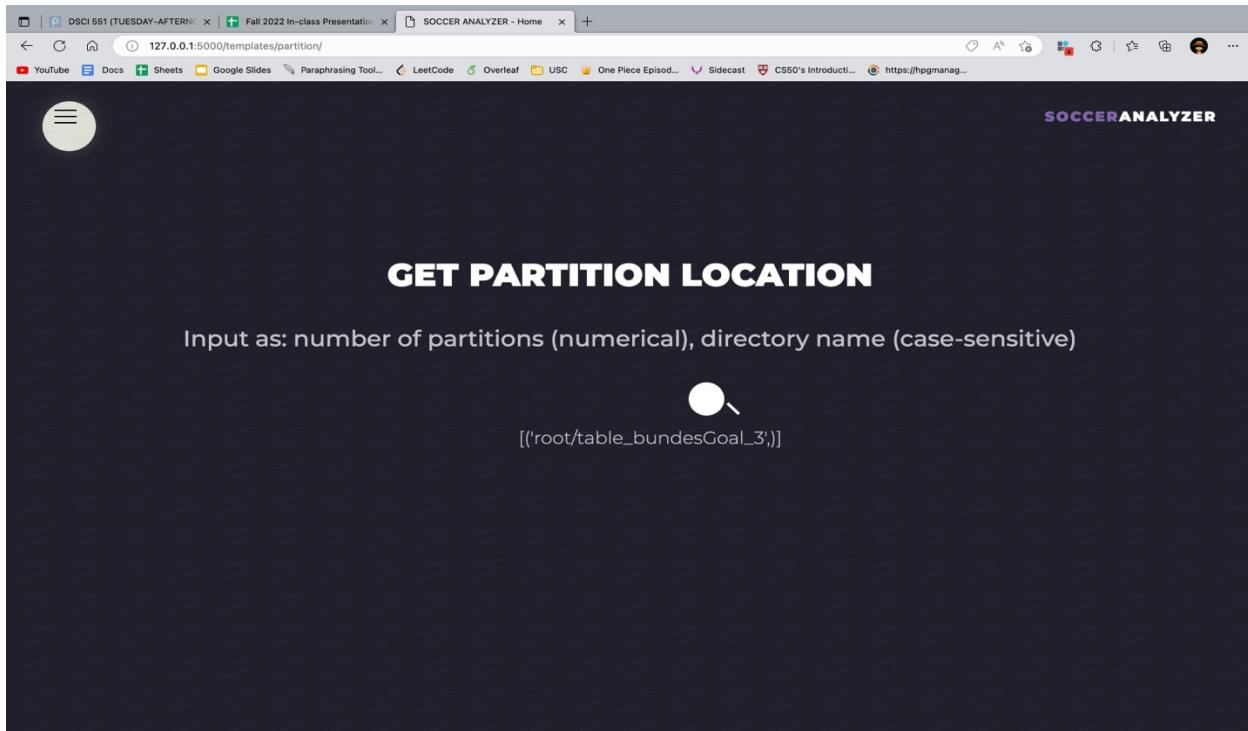


Fig 28. Get partition location function output

ID	player_name	gameTime	goals	xG	assists	xA	shots	key_passes	yellow_cards	red_cards	pos	team	npg	npXG	xGChain	xBuildup
2676	Erik Thommy	8	312	0	0.4598511829972267	2	0.8155245874077082	12	5	0	0	D F M S	0	0.4598511829972267	1.4215824529528618	0.3527394831180572
2691	Milos Veljkovic	23	1723	0	0.6417456492781639	0	0.0274531785396721	3	1	1	0	D S M S	0	0.6417456492781639	2.098258198238909	2.003644482806623
3059	Axel Witsel	15	1130	0	0.5135512463748455	0	0.366133950650692	12	4	4	0	M S M S	0	0.5135512463748455	6.03260762616992	5.50047143916044
3286	Kevin Trapp	33	2970	0	0.0	0	0.0	0	0	0	0	GK D	0	0.0	3.40783329680562	3.40783329680562
3290	Benjamin Stambouli	24	1961	0	0.0428114049136638	2	0.6104596108198166	2	8	5	0	Schalke 04 M S	0	0.0428114049136638	3.3181262593716383	2.79693187214341
3339	Jérôme Roussillon	20	1236	0	0.8446733858436346	2	1.9913508808240292	10	16	0	0	D S M S	0	0.8446733858436346	5.678017353639007	3.663169940933585
3363	Benjamin Pavard	24	1945	0	0.2527241464704275	0	1.4815785773098469	10	13	3	0	D S M S	0	0.2527241464704275	10.882865178398788	9.449053923599422
3393	Bouna Sarr	8	452	0	0.0534978387877345	1	0.074988005682826	3	3	1	0	D S M S	0	0.0534978387877345	1.238306721920967	1.147236950695515
3834	Ishak Belfodil	14	601	0	1.485704567283392	1	0.6725003831088543	11	7	0	0	F S F M	0	1.485704567283392	3.24861397216845	1.2448654510080814
4271	Janik Haberer	14	479	0	0.537568393163383	0	0.0855881813913583	10	2	1	0	F M S	0	0.537568393163383	1.8963787266984584	1.288126740604639
4336	Raphael Framberger	21	1549	0	0.016557123361587	1	2.43236887641251	1	11	4	1	D M S Augsburg	0	0.016557123361587	5.058925217017531	3.91922030411607
5086	Marc Roca	6	179	0	0.2796180993319558	0	0.0188390724350942	3	1	1	0	M S M S	0	0.2796180993319558	0.51369122414578102	0.2152340486645698
5133	Aarón Martín	5	259	0	0.0243980269879102	0	0.182888968847692	1	3	0	0	M S M S	0	0.0243980269879102	0.0756111312657594	0.0414297506213188
5238	Jordan Torunarigha	14	971	0	0.1476941518485546	0	0.4518662691116333	4	3	5	0	D S M S	0	0.1476941518485546	2.820356484502554	2.478442575782537
5241	Péter Gulácsi	33	2970	0	0.0	0	0.0	0	0	0	0	GK RasenBallsport Leipzig	0	0.0	5.1437314273789525	5.1437314273789525
5253	Beno Schmitz	12	405	0	0.2080057952553033	1	0.2649602983146906	4	5	0	0	D S FC Cologne	0	0.2080057952553033	1.655946284532547	1.432701986283064
5260	Andreas Luthe	31	2770	0	0.0	0	0.0	0	0	1	0	GK Union Berlin	0	0.0	2.496423464268446	2.496423464268446
5262	Alexander Schwolow	26	2340	0	0.0	0	0.0	0	0	0	0	GK Hertha Berlin	0	0.0	3.677420792169869	3.677420792169869
5263	Lukas Kühler	19	884	0	0.0805985620245337	1	0.4371448513120413	2	5	2	0	D S Freiburg	0	0.0805985620245337	1.8345615938305853	1.4810890089720488

Fig 29. Read partition output

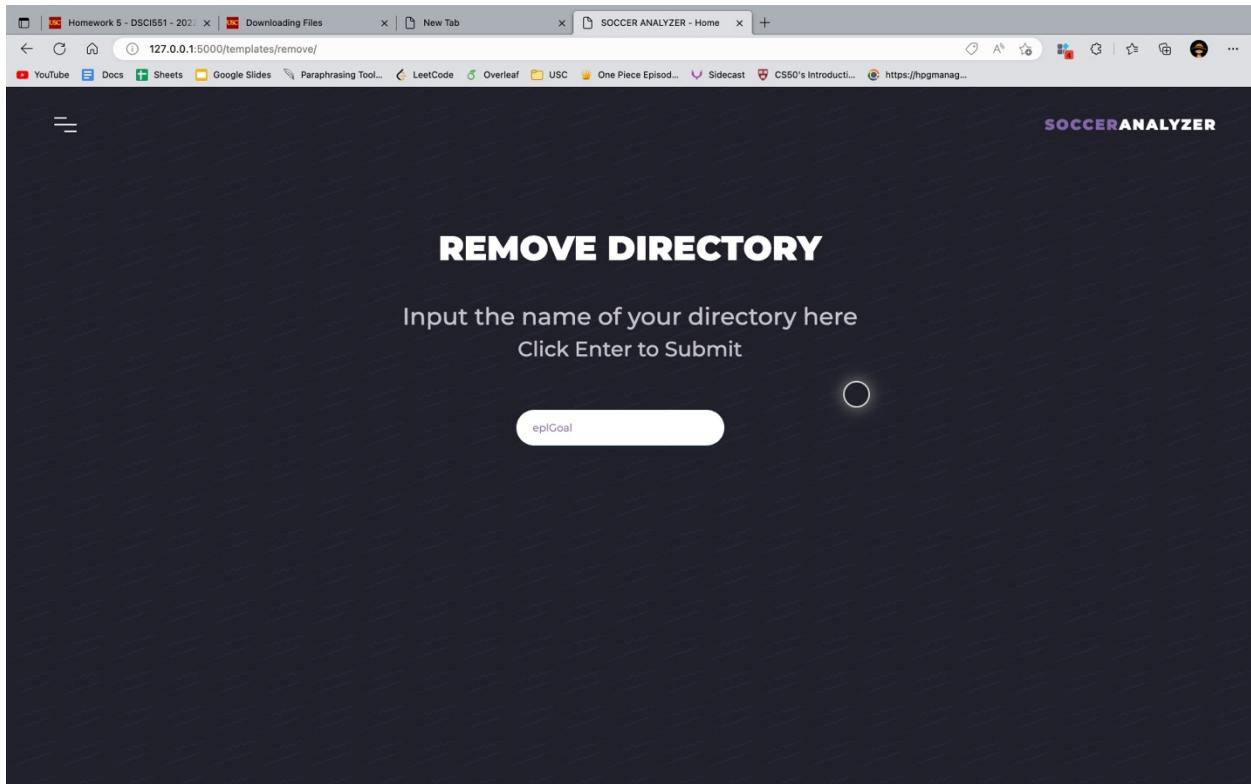


Fig 30. Remove directory

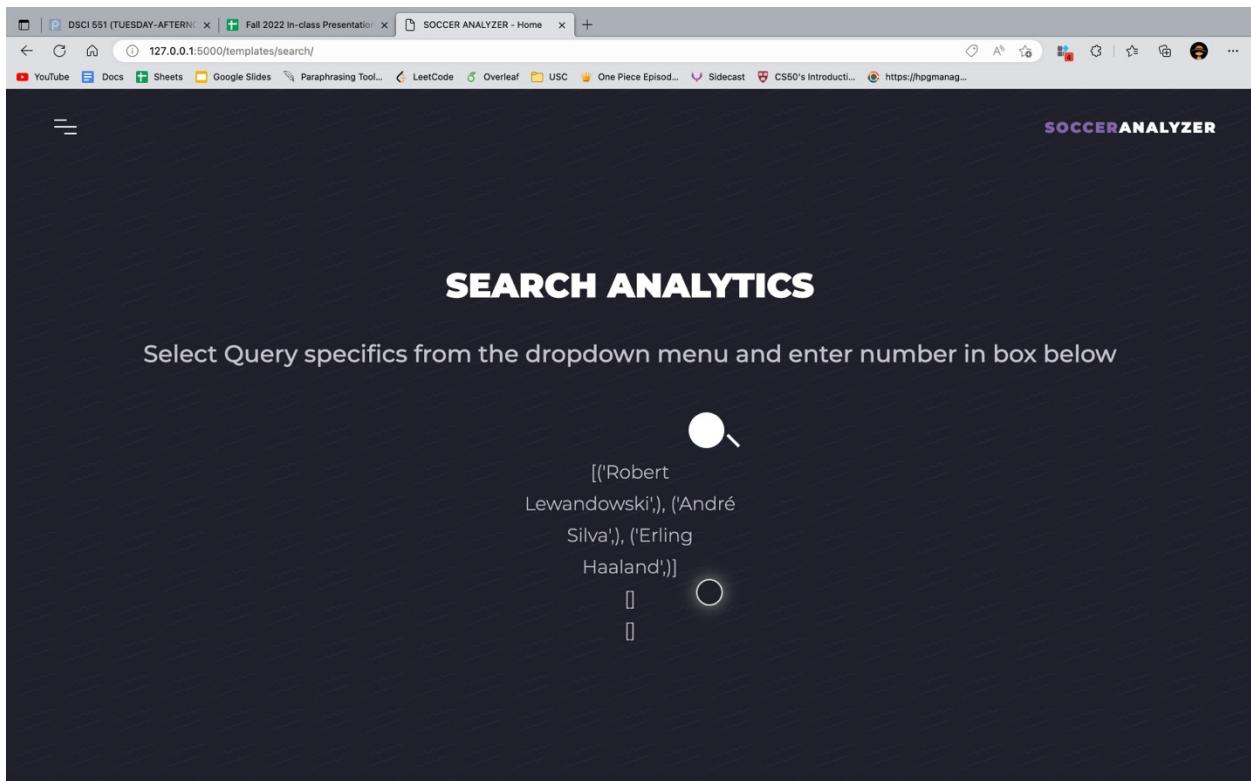


Fig 31. Search analytics

MongoDB and SQL Comparison:

- MongoDB works well with unstructured data while SQL can only handle structured data.
- MongoDB is more flexible and scalable as compared to SQL
- MongoDB is faster and works well when the amount of data is large compared to SQL
- SQL is developer friendly. MongoDB cannot support traditional SQL queries the way SQL does.
- SQL is more suitable when the data is structure and when there is a need for traditional relational database.

In this case and then the data stored in name node is structured. As the data sets are CSV files, the data to be stored in the data node is also organized. The size of the dataset is not very large. Hence SQL offers a more convenient way of implementing emulated HDFS. The resources are well documented which makes the implementation easier.

Development experience:

Developing this project provided a deeper insight into the working of HDFS file system. It gave a better understanding of the role played by the name node and data node in the file system. We could also explore command execution and querying using MongoDB and SQL databases through python scripts. The User interface was made as interpretable as possible while all the complex query execution happened in the back end. MongoDB and SQL have plenty of resources and documentation online which made the implementation easier.

Link of the Code and Demo:

https://drive.google.com/drive/folders/1NRfcEKBHVSEvRTEbzmx9j95LeUsE7Tor?usp=share_link

[Soccer Analytics\(EDFS\) - YouTube](#)