Exploring the Career Performances of ODI Cricketers

Big Data: 44517-04

Hadoop Hackers

Anilkumar Palavelli, Bhanuteja Chitrala, Bala Harinadh Palavelli, Krishna Sai Balupari, Samyuktha Pandula, Suresh Sindam, RajaShree Bokka

GitHub Link: https://github.com/Bhanuteja009/CricketersODIStatsAnalysis

Project Idea:

The focus of this study is on a large dataset that includes all of the career numbers of One Day International (ODI) cricketers from the start of ODI cricket in India to the present day. Important performance metrics are included in the dataset, such as the number of games played, innings finished, runs scored, batting averages, highest scores (not outs), bowling data, and the length of each player's career. We want to find important trends, accomplishments, and success milestones in Indian ODI cricket by carefully analyzing and visualizing this dataset. Many cricket fans and students will find this study very helpful because it shows how Indian ODI players have contributed to the country's cricketing history and how they still do so today.

Technology Summary:

Visual Studio Code (VS Code):

IDE (Integrated Development Environment): VS Code will serve as the primary environment for coding, debugging, and managing the project. It provides various extensions and tools that support Java development.

Java:

Programming Language: Java will be used to process and analyze the large dataset. Java offers robust libraries and functionalities for handling data processing and manipulation.

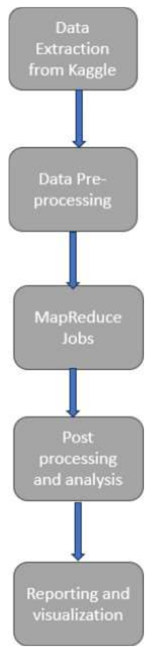
Hadoop MapReduce:

Distributed Computing Framework: Hadoop MapReduce will be employed to handle the large dataset. This framework is particularly efficient for processing vast amounts of data by distributing the workload across multiple nodes in a cluster.

Tableau:

Data Visualization Tool: Tableau will be utilized for creating visual representations of the analyzed data. It offers a user-friendly interface to generate various visualizations like charts, graphs, and dashboards, enabling the effective communication of insights derived from the dataset.

Architecture Diagram:


Data Extraction from Kaggle
Data Pre-processing
MapReduce Jobs
Post processing and analysis
Reporting and visualization

Architecture Summary:

Data Extraction:
Export the data from Excel to a format that Hadoop MapReduce can process. (excel)
Ensure that the data is structured properly, with appropriate headers and consistent formatting.

Data Preprocessing (if necessary):
Data preprocessing may involve cleaning, transforming, and filtering the data to prepare it for MapReduce processing.
Common preprocessing tasks include handling missing values, removing duplicates, and aggregating data.

MapReduce Jobs:
Develop MapReduce programs using Java.
Create two main functions: a Mapper and a Reducer. The Mapper extracts and emits key-value pairs, while the Reducer processes and aggregates the data.
The output is typically key-value pairs, where keys represent categories or identifiers, and values represent calculated results.

Post-processing and Analysis: If necessary, perform additional post-processing or analysis on the MapReduce results to generate insights or visualizations.

Reporting and Visualization:
Utilize data visualization tools like Tableau, or other reporting tools, to present the calculated results in a meaningful and understandable way**.**

Goals:

Goal 1: The goal is to visualize the players who are good at both batting and bowling, that is regular all-rounders (whose bowling and batting average greater than 40).

Goal 2: The goal is to visualize all the bowlers who bowled more than 150 maiden overs.

Goal 3: To visualize all the players who scored more than 20 centuries.

Goal 4: To visualize all the players who took more than 250 wickets in ODI cricket.

Goal 5: To visualize all the players whose took more than 145 catches.

Goal 6: The goal is to visualize all the players who remained not out in more than 50 matches.

Project Description:

1. Project Setup:

Installation of the environment:

Install VS Code and Java Extensions: VS Code and any necessary Java modules should be installed first. When you're writing code, fixing bugs, and managing projects, use VS Code as your main Integrated Development Environment (IDE). To allow Java development without any problems, make sure that Java features are added.

Double-check the Java environment: Make sure that Java is installed and set up correctly on your computer. Make sure the system environment variables are set correctly and that the Java development kit (JDK) is added.

Access to the dataset and its format:

Check if Dataset Exists: Make sure that the ODI players' dataset is available. Make sure the information includes all of India's ODI records and includes success measures like games played, runs made, hitting rates, bowling stats, and so on.

Check the Structure of the Dataset: Make sure that the dataset is organized properly, with the right titles and uniform style. If you need to, clean and organize the information so that the next steps go smoothly.

2. Data Extraction:

Use VS Code to write a Java program:

Make a new Java file in Visual Studio Code (VS Code).

To get data from the Excel file, you can use Java. For Excel tasks, you can use tools like Apache POI.

Export data in a format that works with Hadoop MapReduce:

Once the data has been read in Java, it needs to be exported in a manner that Hadoop MapReduce can use. Usually, this means changing the data into an organized file that can be processed by MapReduce or into key-value pairs.

Save the files that have been handled in a standard format that MapReduce tools can easily read, such as CSV (Comma-Separated Values).

3. Data Preprocessing:

Handling Missing Values: Check for any missing or null values in the dataset. If found, decide on an appropriate strategy to handle them, such as imputation or removal.

Removing Duplicates: Identify and remove any duplicate records from the dataset to ensure data integrity.

Data Filtering: Apply filters to extract relevant subsets of data based on the project goals, such as players with more than 150 matches, bowlers with more than 40 maiden overs, etc.

4. Hadoop MapReduce:

Developed MapReduce Programs in Java: Develop MapReduce programs in Java to process the preprocessed data.

Created Mapper Function: Create a Mapper function to extract and emit key-value pairs based on the analysis goals.

Implemented Reducer Function: Implement a Reducer function to process and aggregate the data.

Output Format for Further Analysis: Ensure the output is in a format suitable for further analysis and visualization.

5. Post-processing and Analysis:

Analyzed MapReduce Results: Thoroughly analyze the results generated by the MapReduce jobs, uncovering intricate patterns, trends, and actionable insights in the processed data.

Identified Patterns and Trends: Identify patterns, trends, and insights in the processed data.

Performed Additional Calculations and Filtering: Perform any additional calculations or filtering based on project requirements.

6. Reporting and Visualization:

Using Tableau:

Save the processed data in a file format compatible with Tableau, such as CSV.

Connect to the saved data in Tableau: Import the data into Tableau for visualization.

Create visuals:

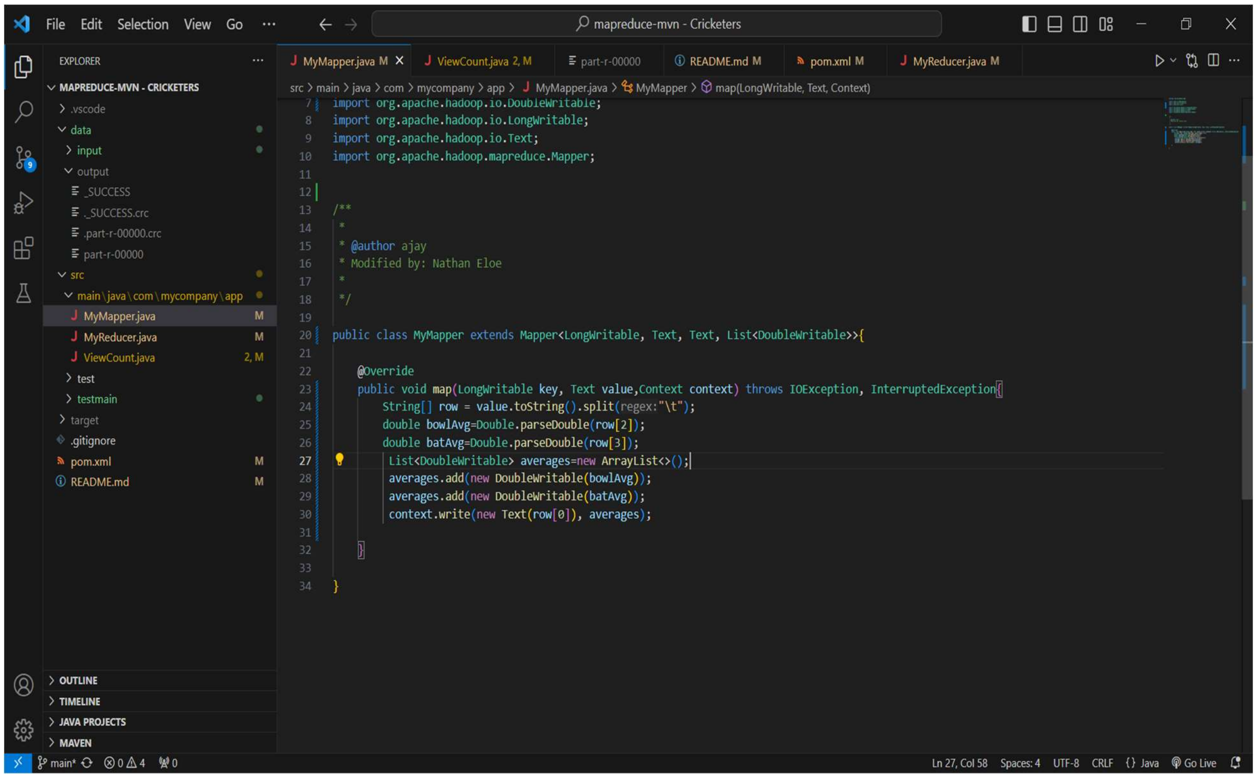
Design visualizations based on project goals.

Generate charts for players with more than 150 matches, bowlers with more than 40 maiden overs, all-rounders, etc.

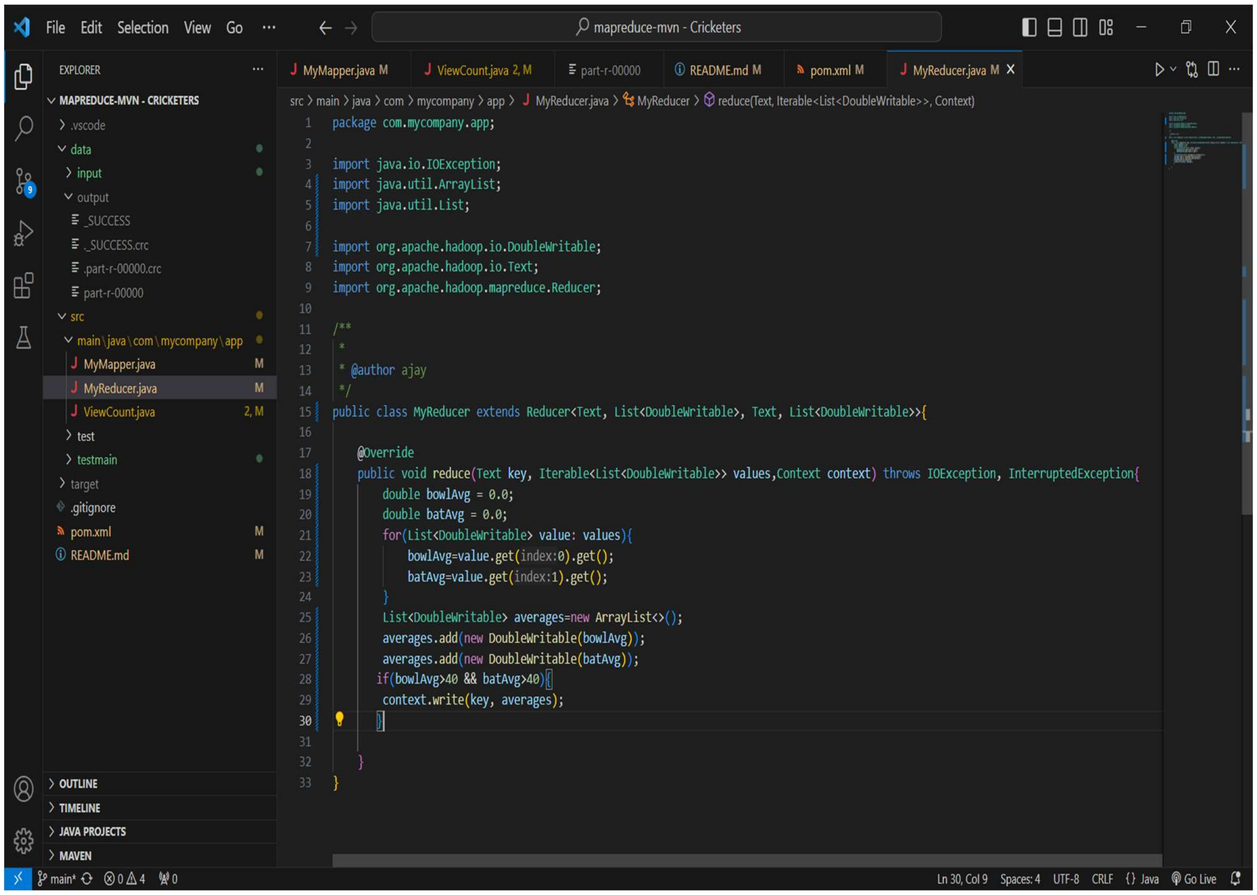
Use Tableau's features to create interactive dashboards and insightful visual representations.

Results Summary:

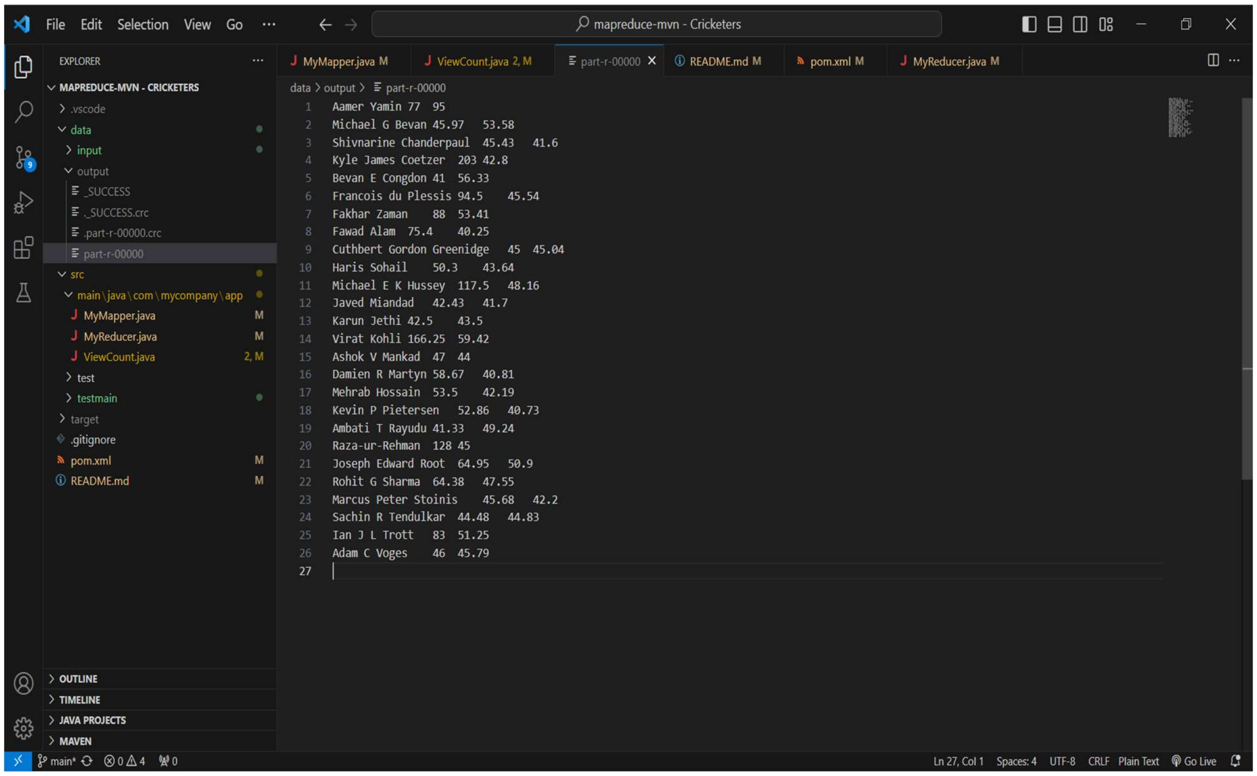
1. **Goal 1: The goal is to visualize the players who are good at both batting and bowling, that is regular all-rounders (whose bowling and batting average greater than 40).**



```java
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;


/**
 *
 * @author ajay
 * Modified by: Nathan Eloe
 *
 */

public class MyMapper extends Mapper<LongWritable, Text, Text, List<DoubleWritable>>{

    @Override
    public void map(LongWritable key, Text value,Context context) throws IOException, InterruptedException{
        String[] row = value.toString().split(regex:"\t");
        double bowlAvg=Double.parseDouble(row[2]);
        double batAvg=Double.parseDouble(row[3]);
        List<DoubleWritable> averages=new ArrayList<>();
        averages.add(new DoubleWritable(bowlAvg));
        averages.add(new DoubleWritable(batAvg));
        context.write(new Text(row[0]), averages);

    }

}
```



```java
package com.mycompany.app;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

/**
 *
 * @author ajay
 */
public class MyReducer extends Reducer<Text, List<DoubleWritable>, Text, List<DoubleWritable>>{

    @Override
    public void reduce(Text key, Iterable<List<DoubleWritable>> values,Context context) throws IOException, InterruptedException{
        double bowlAvg = 0.0;
        double batAvg = 0.0;
        for(List<DoubleWritable> value: values){
            bowlAvg=value.get(index:0).get();
            batAvg=value.get(index:1).get();
        }
        List<DoubleWritable> averages=new ArrayList<>();
        averages.add(new DoubleWritable(bowlAvg));
        averages.add(new DoubleWritable(batAvg));
       if(bowlAvg>40 && batAvg>40){
        context.write(key, averages);
       }

    }
}
```


File Edit Selection View Go
mapreduce-mvn - Cricketers
EXPLORER
MAPREDUCE-MVN - CRICKETERS
MyMapper.java M
ViewCount.java 2, M
part-r-00000
README.md M
pom.xml M
MyReducer.java M
data > output > part-r-00000
main* 0 4 0
Ln 27, Col 1 Spaces: 4 UTF-8 CRLF Plain Text Go Live

```
Aamer Yamin 77  95
Michael G Bevan 45.97   53.58
Shivnarine Chanderpaul  45.43   41.6
Kyle James Coetzer  203 42.8
Bevan E Congdon 41  56.33
Francois du Plessis 94.5    45.54
Fakhar Zaman    88  53.41
Fawad Alam  75.4    40.25
Cuthbert Gordon Greenidge   45  45.04
Haris Sohail    50.3    43.64
Michael E K Hussey  117.5   48.16
Javed Miandad   42.43   41.7
Karun Jethi 42.5    43.5
Virat Kohli 166.25  59.42
Ashok V Mankad  47  44
Damien R Martyn 58.67   40.81
Mehrab Hossain  53.5    42.19
Kevin P Pietersen   52.86   40.73
Ambati T Rayudu 41.33   49.24
Raza-ur-Rehman  128 45
Joseph Edward Root  64.95   50.9
Rohit G Sharma  64.38   47.55
Marcus Peter Stoinis    45.68   42.2
Sachin R Tendulkar  44.48   44.83
Ian J L Trott   83  51.25
Adam C Voges    46  45.79

```

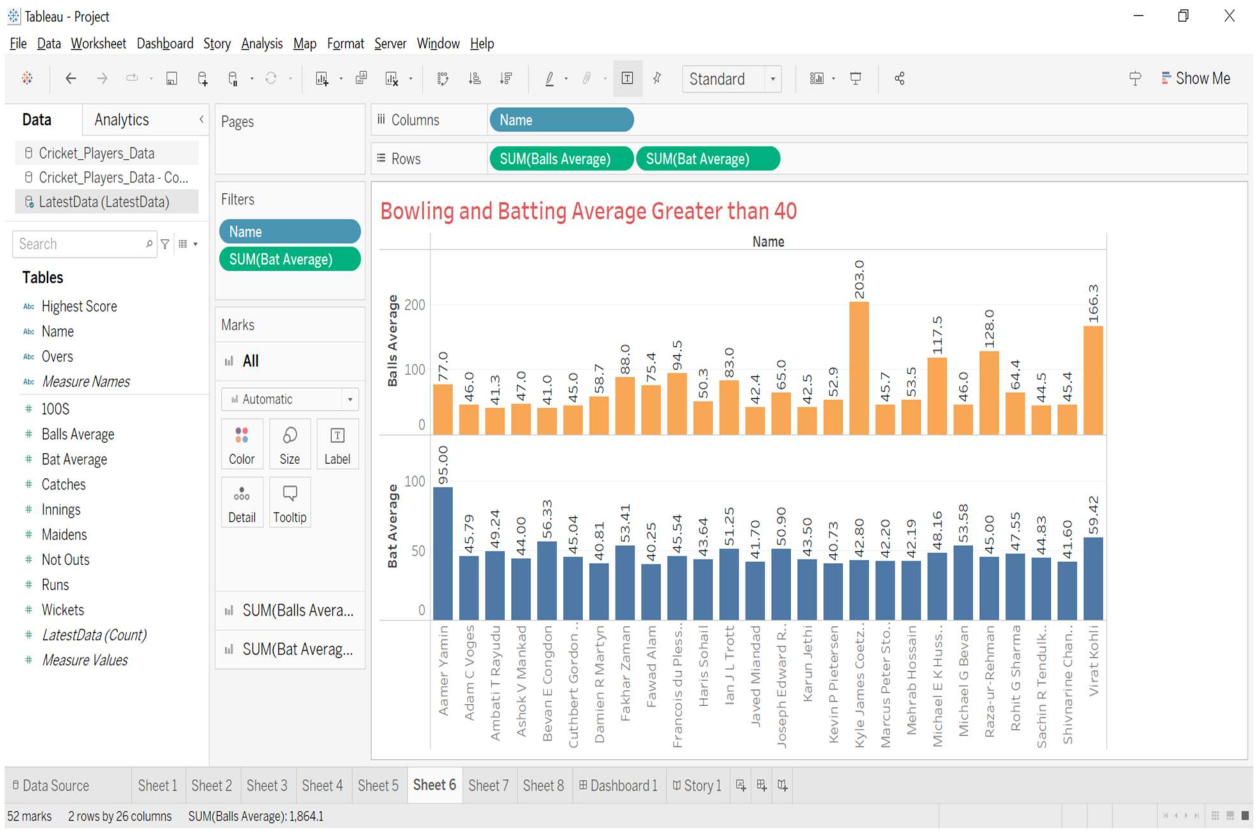


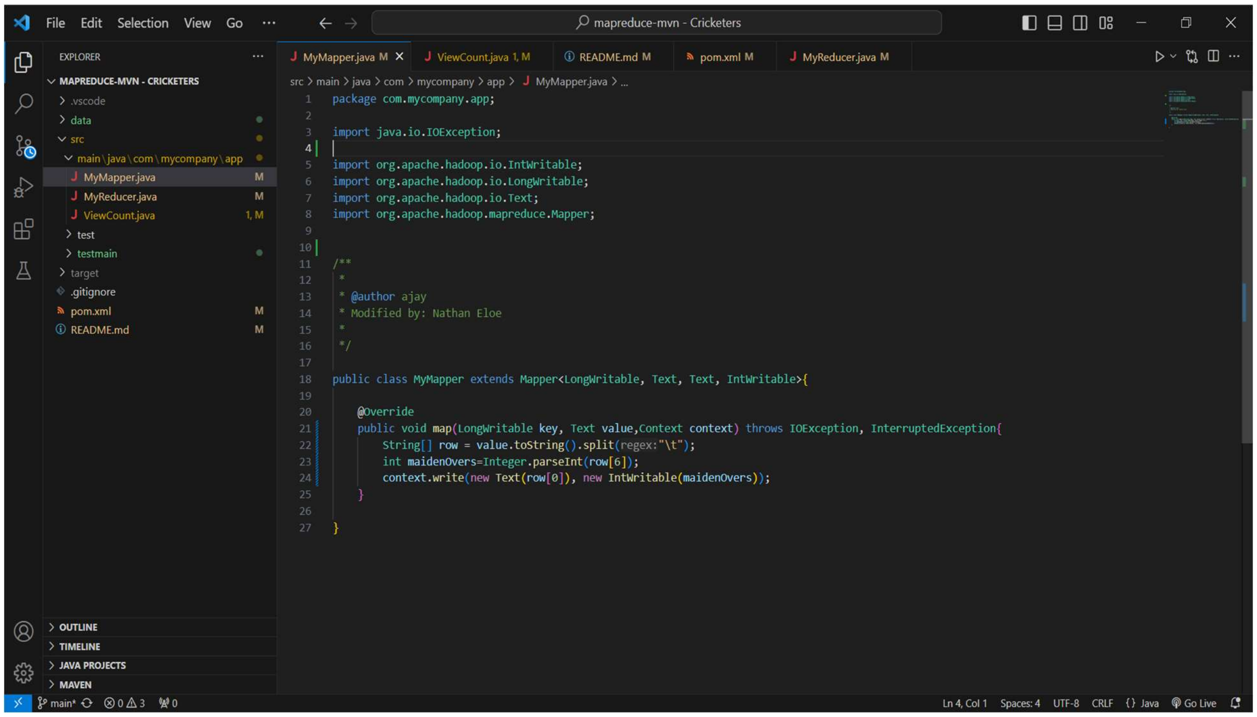
Tableau - Project
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Standard
Show Me
Data Analytics
Cricket_Players_Data
Cricket_Players_Data - Co...
LatestData (LatestData)
Search
Tables
Highest Score
Name
Overs
Measure Names
100S
Balls Average
Bat Average
Catches
Innings
Maidens
Not Outs
Runs
Wickets
LatestData (Count)
Measure Values
Pages
Filters
Name
SUM(Bat Average)
Marks
All
Automatic
Color Size Label
Detail Tooltip
SUM(Balls Avera...
SUM(Bat Averag...
Columns Name
Rows SUM(Balls Average) SUM(Bat Average)
Bowling and Batting Average Greater than 40
Name
Balls Average
Bat Average
Aamer Yamin 77.0 95.00
Adam C Voges 46.0 45.79
Ambati T Rayudu 41.3 49.24
Ashok V Mankad 47.0 44.00
Bevan E Congdon 41.0 56.33
Cuthbert Gordon .. 45.0 45.04
Damien R Martyn 58.7 40.81
Fakhar Zaman 88.0 53.41
Fawad Alam 75.4 40.25
Francois du Pless.. 94.5 45.54
Haris Sohail 50.3 43.64
Ian J L Trott 83.0 51.25
Javed Miandad 42.4 41.70
Joseph Edward R.. 65.0 50.90
Karun Jethi 42.5 43.50
Kevin P Pietersen 52.9 40.73
Kyle James Coetz.. 203.0 42.80
Marcus Peter Sto.. 45.7 42.20
Mehrab Hossain 53.5 42.19
Michael E K Huss.. 117.5 48.16
Michael G Bevan 46.0 53.58
Raza-ur-Rehman 128.0 45.00
Rohit G Sharma 64.4 47.55
Sachin R Tendulk.. 44.5 44.83
Shivnarine Chan.. 45.4 41.60
Virat Kohli 166.3 59.42
Data Source Sheet 1 Sheet 2 Sheet 3 Sheet 4 Sheet 5 Sheet 6 Sheet 7 Sheet 8 Dashboard 1 Story 1
52 marks 2 rows by 26 columns SUM(Balls Average): 1,864.1

Story:

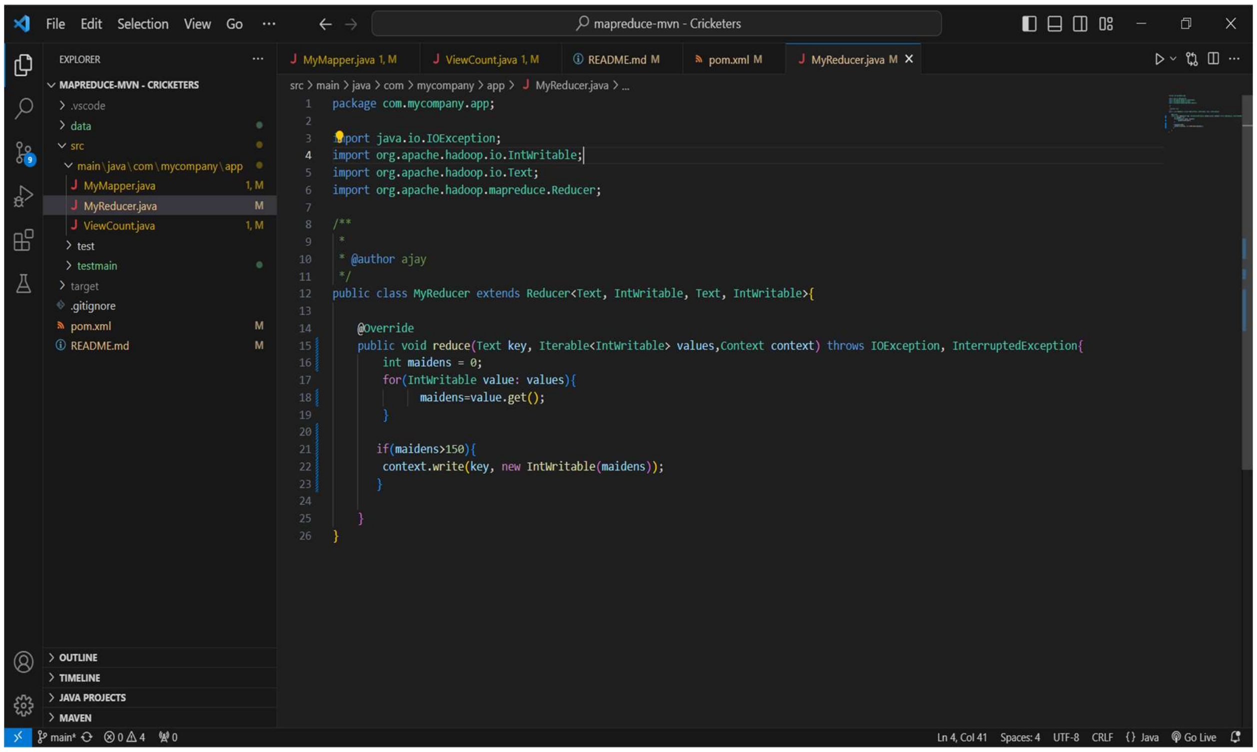
This data features cricket players with both batting and bowling averages exceeding 40, showcasing their versatile skills in both aspects of the game. Notable performers include Virat Kohli, Joseph Edward Root, and Rohit G Sharma, who maintain high averages in both batting and bowling.

Aamer Yamin, though primarily recognized for his bowling, demonstrates commendable skills with a batting average of 95. The list encompasses a diverse range of players like Shivnarine Chanderpaul, Michael Hussey, and Sachin Tendulkar, highlighting their balanced contributions to both batting and bowling departments in the world of cricket.

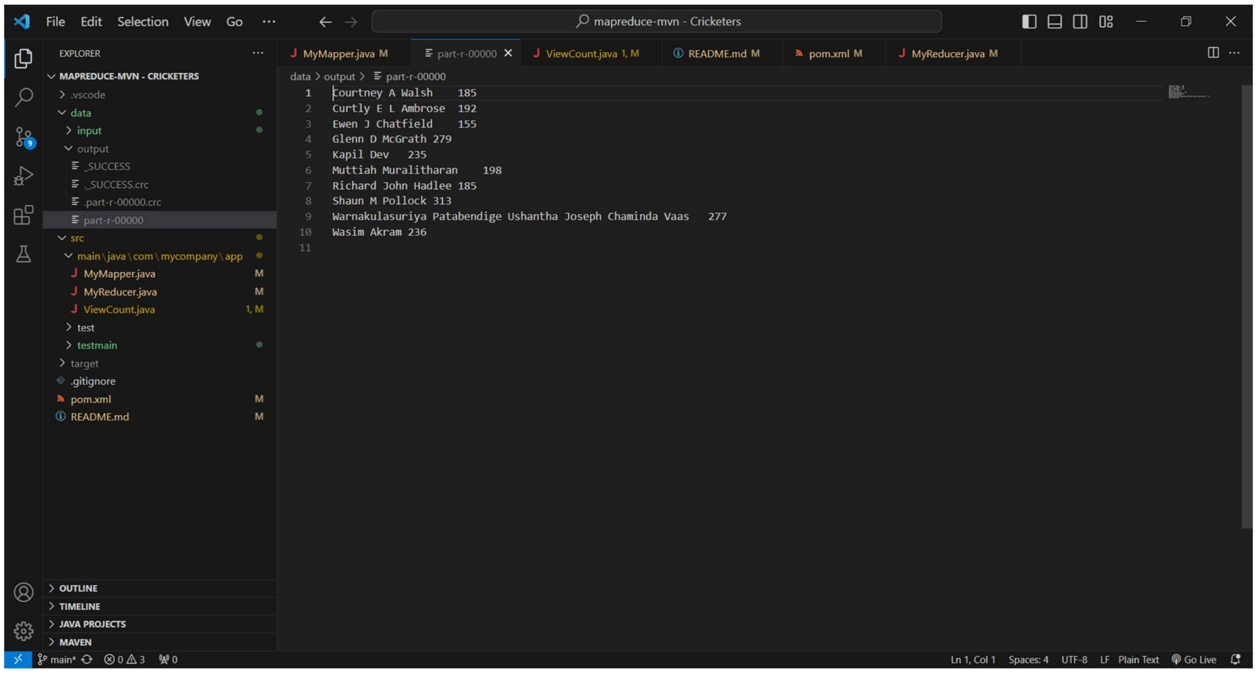
2. Goal 2: The goal is to visualize all the bowlers who bowled more than 150 maiden overs.



```java
package com.mycompany.app;

import java.io.IOException;


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


/**
 *
 * @author ajay
 * Modified by: Nathan Eloe
 *
 */

public class MyMapper extends Mapper<LongWritable, Text, Text, IntWritable>{

    @Override
    public void map(LongWritable key, Text value,Context context) throws IOException, InterruptedException{
        String[] row = value.toString().split(regex:"\t");
        int maidenOvers=Integer.parseInt(row[6]);
        context.write(new Text(row[0]), new IntWritable(maidenOvers));
    }

}
```

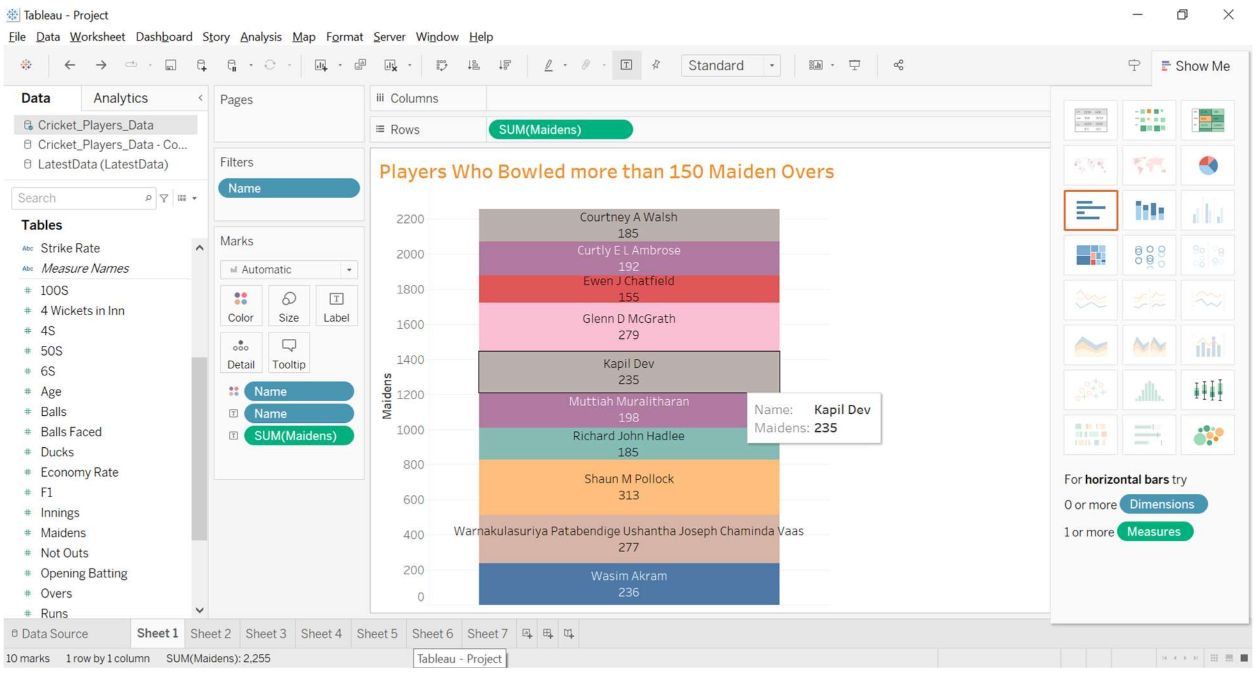


```java
package com.mycompany.app;

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

/**
 *
 * @author ajay
 */
public class MyReducer extends Reducer<Text, IntWritable, Text, IntWritable>{

    @Override
    public void reduce(Text key, Iterable<IntWritable> values,Context context) throws IOException, InterruptedException{
        int maidens = 0;
        for(IntWritable value: values){
              maidens=value.get();
        }

       if(maidens>150){
        context.write(key, new IntWritable(maidens));
       }

    }
}
```



data > output > part-r-00000

```
Courtney A Walsh	185
Curtly E L Ambrose	192
Ewen J Chatfield	155
Glenn D McGrath	279
Kapil Dev	235
Muttiah Muralitharan	198
Richard John Hadlee	185
Shaun M Pollock	313
Warnakulasuriya Patabendige Ushantha Joseph Chaminda Vaas	277
Wasim Akram	236
```



Players Who Bowled more than 150 Maiden Overs

Rows: SUM(Maidens)

Filters: Name

10 marks   1 row by 1 column   SUM(Maidens): 2,255

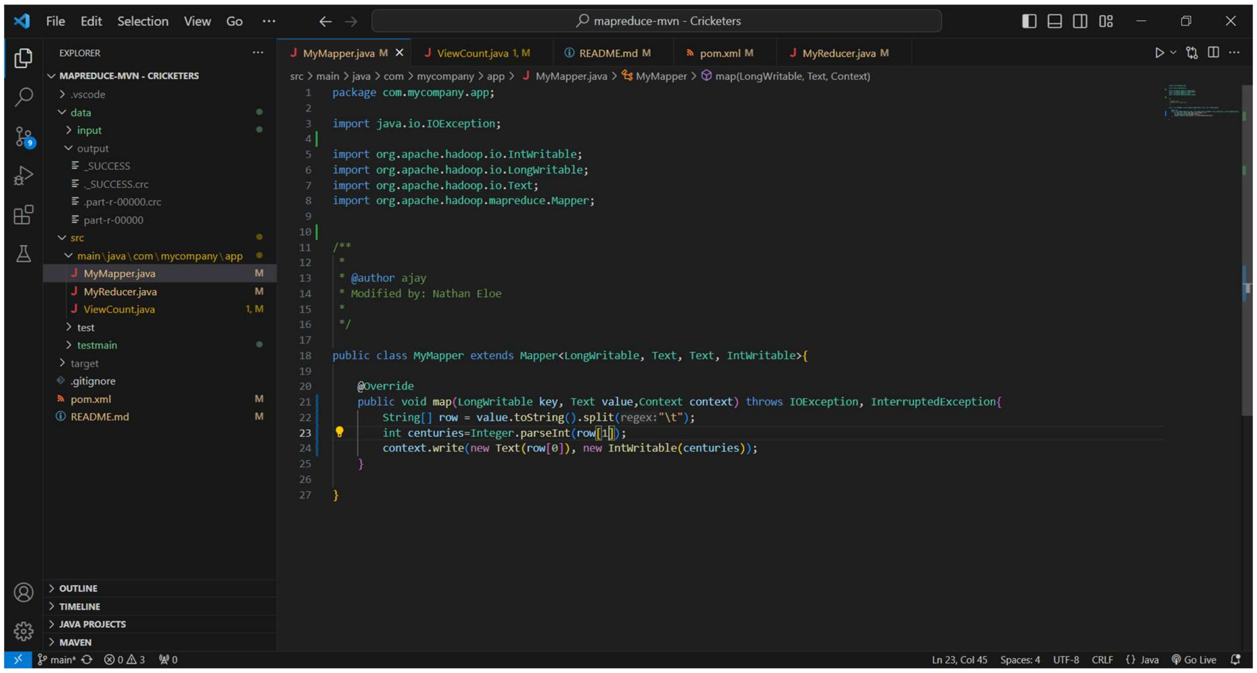
Story:

In the illustrious realm of cricket, a distinguished group of bowlers emerges as masters of control and strategy, having played more than 150 maiden overs. Led by Shaun M Pollock with an extraordinary 313 maiden overs, the list includes iconic names such as Glenn D McGrath, Kapil Dev, and Muttiah Muralitharan, each contributing their unique blend of skill and consistency.

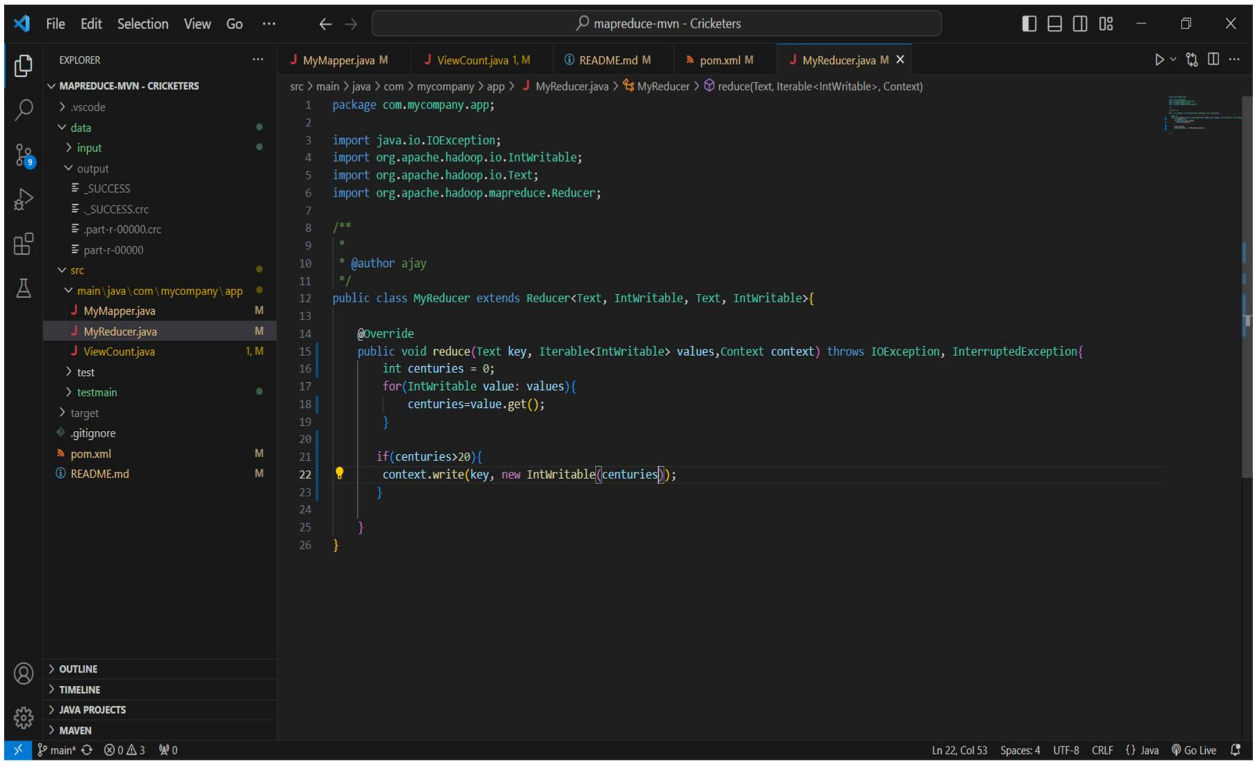
The renowned duo of Courtney A Walsh and Richard John Hadlee shares a mark of 185 maiden overs, showcasing enduring excellence. Warnakulasuriya Patabendige Ushantha Joseph

Chaminda Vaas and Wasim Akram add to this elite circle, with figures of 277 and 236 maiden overs, leaving an indelible imprint on the cricketing landscape.

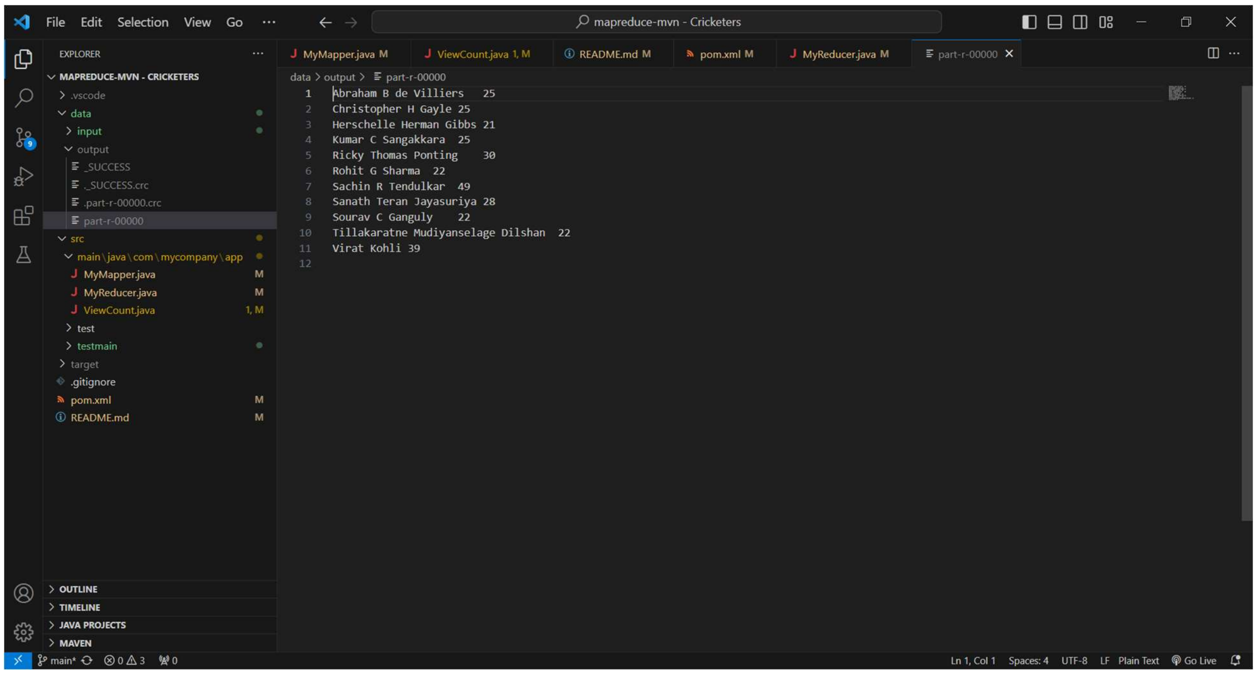
3. Goal 3: To visualize all the players who scored more than 20 centuries.



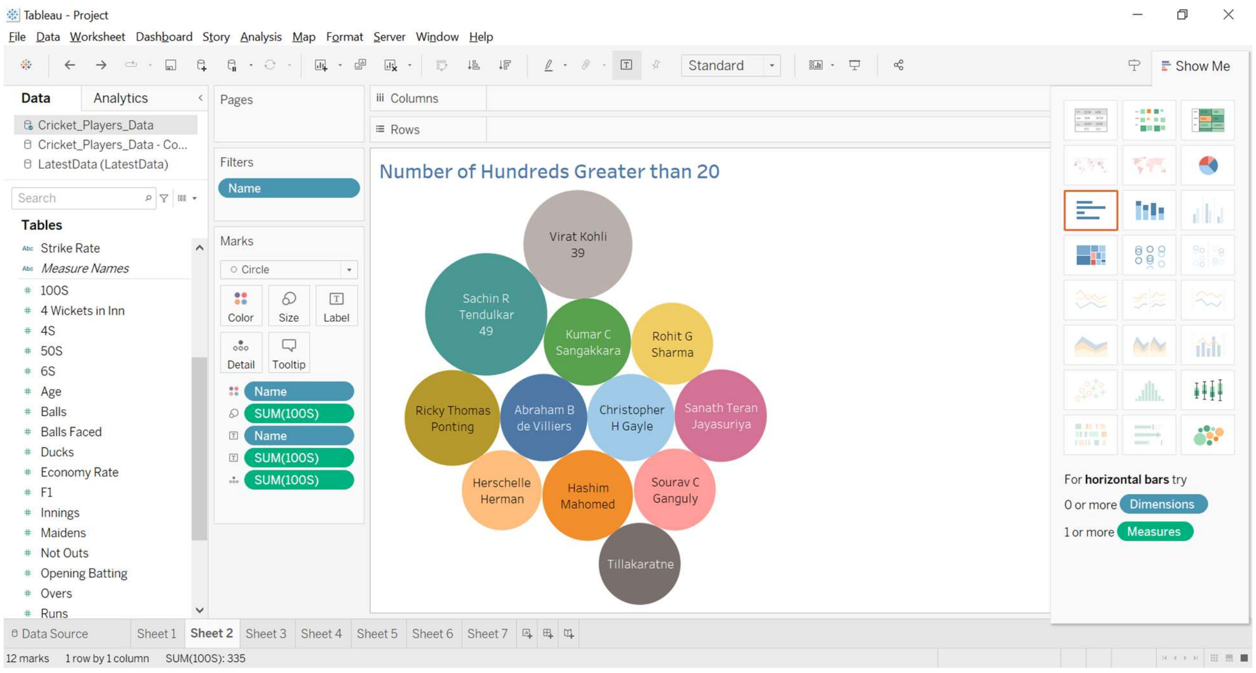
```java
package com.mycompany.app;

import java.io.IOException;

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


/**
 *
 * @author ajay
 * Modified by: Nathan Eloe
 *
 */

public class MyMapper extends Mapper<LongWritable, Text, Text, IntWritable>{

    @Override
    public void map(LongWritable key, Text value,Context context) throws IOException, InterruptedException{
        String[] row = value.toString().split(regex:"\t");
        int centuries=Integer.parseInt(row[1]);
        context.write(new Text(row[0]), new IntWritable(centuries));
    }

}
```



```java
package com.mycompany.app;

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

/**
 *
 * @author ajay
 */
public class MyReducer extends Reducer<Text, IntWritable, Text, IntWritable>{

    @Override
    public void reduce(Text key, Iterable<IntWritable> values,Context context) throws IOException, InterruptedException{
        int centuries = 0;
        for(IntWritable value: values){
            centuries=value.get();
        }

       if(centuries>20){
        context.write(key, new IntWritable(centuries));
       }

    }
}
```


mapreduce-mvn - Cricketers
EXPLORER
MAPREDUCE-MVN - CRICKETERS
MyMapper.java
ViewCount.java
README.md
pom.xml
MyReducer.java
part-r-00000
data > output > part-r-00000

```
Abraham B de Villiers	25
Christopher H Gayle	25
Herschelle Herman Gibbs	21
Kumar C Sangakkara	25
Ricky Thomas Ponting	30
Rohit G Sharma	22
Sachin R Tendulkar	49
Sanath Teran Jayasuriya	28
Sourav C Ganguly	22
Tillakaratne Mudiyanselage Dilshan	22
Virat Kohli	39
```

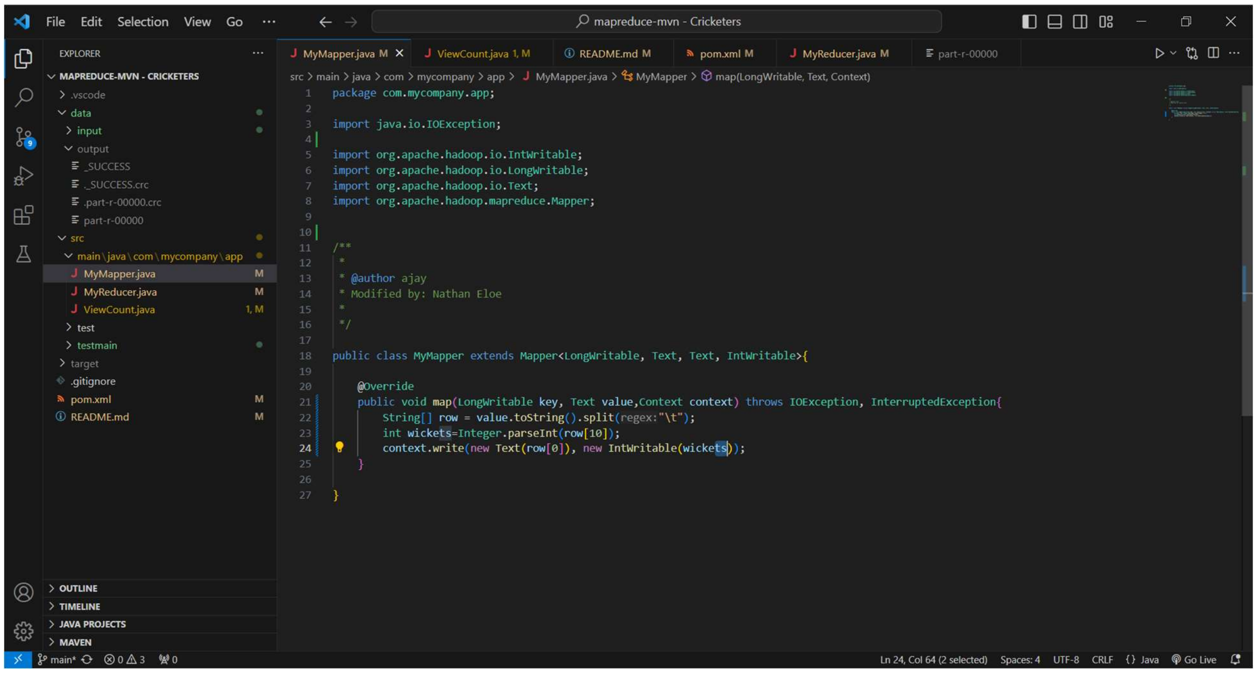


Story:

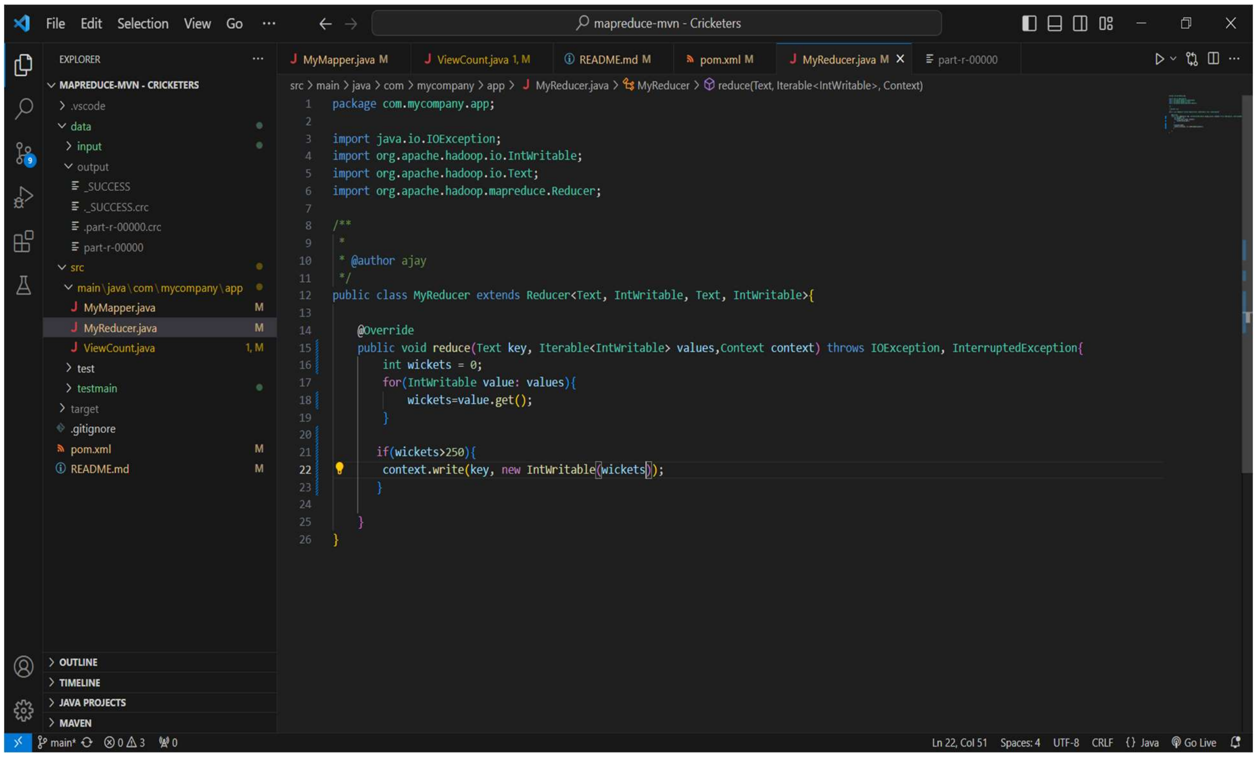
In the illustrious realm of cricket centurions, a stellar cast of players has showcased remarkable mastery, each having scored more than 20 centuries. Virat Kohli leads the pack with an astonishing 39 centuries, standing tall alongside legends such as Sachin Tendulkar with 49 centuries.

The dynamic duo of Abraham B de Villiers and Christopher H Gayle share the glory with 25 centuries each, while stalwarts like Ricky Ponting, Sanath Jayasuriya, and Kumar Sangakkara contribute to this elite circle. Herschelle Gibbs, Rohit Sharma, Sourav Ganguly, and Tillakaratne Dilshan complete the ensemble, each boasting a legacy of century-making excellence in the cricketing arena.

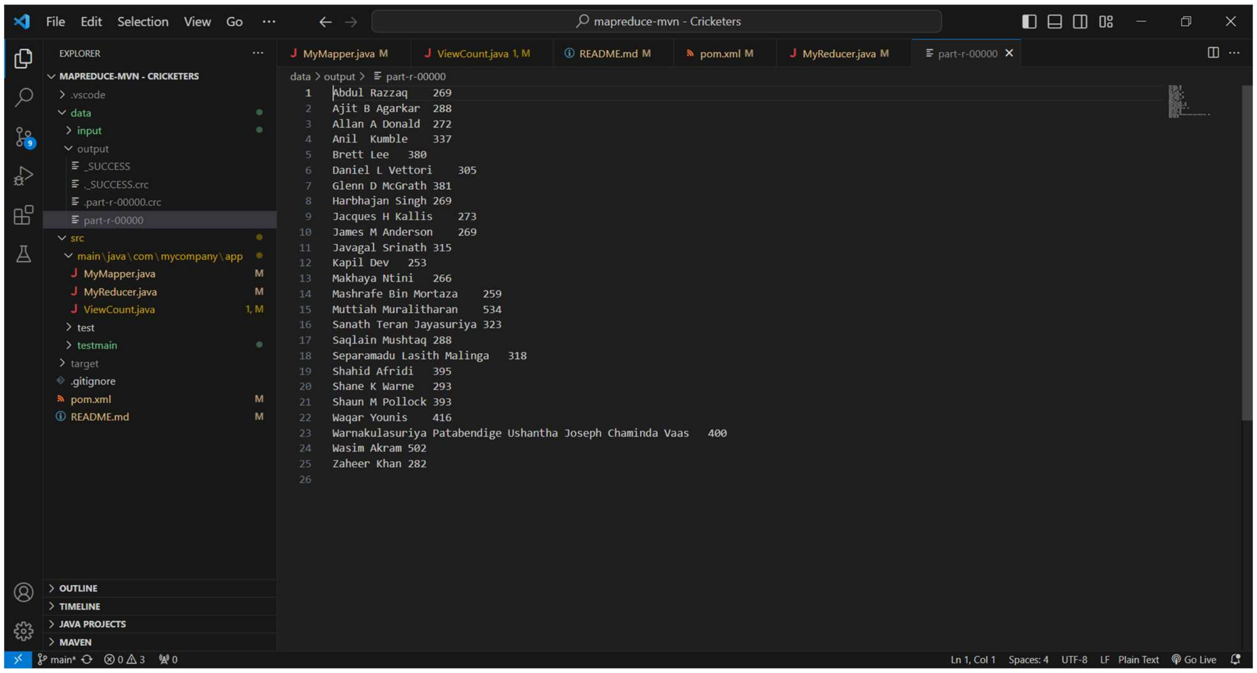
4. Goal 4: To visualize all the players who took more than 250 wickets in ODI cricket.



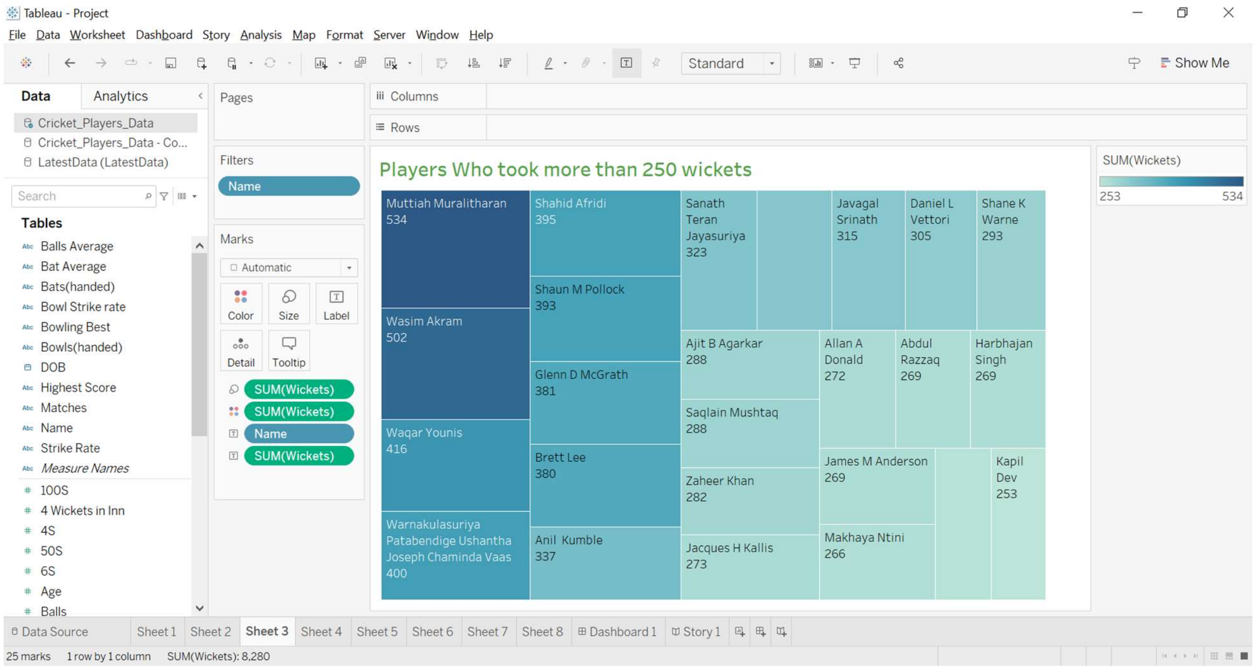
```java
package com.mycompany.app;

import java.io.IOException;

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


/**
 *
 * @author ajay
 * Modified by: Nathan Eloe
 *
 */

public class MyMapper extends Mapper<LongWritable, Text, Text, IntWritable>{

    @Override
    public void map(LongWritable key, Text value,Context context) throws IOException, InterruptedException{
        String[] row = value.toString().split(regex:"\t");
        int wickets=Integer.parseInt(row[10]);
        context.write(new Text(row[0]), new IntWritable(wickets));
    }

}
```



```java
package com.mycompany.app;

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

/**
 *
 * @author ajay
 */
public class MyReducer extends Reducer<Text, IntWritable, Text, IntWritable>{

    @Override
    public void reduce(Text key, Iterable<IntWritable> values,Context context) throws IOException, InterruptedException{
        int wickets = 0;
        for(IntWritable value: values){
            wickets=value.get();
        }

       if(wickets>250){
        context.write(key, new IntWritable(wickets));
       }

    }
}
```



```
Abdul Razzaq	269
Ajit B Agarkar	288
Allan A Donald	272
Anil Kumble	337
Brett Lee	380
Daniel L Vettori	305
Glenn D McGrath	381
Harbhajan Singh	269
Jacques H Kallis	273
James M Anderson	269
Javagal Srinath	315
Kapil Dev	253
Makhaya Ntini	266
Mashrafe Bin Mortaza	259
Muttiah Muralitharan	534
Sanath Teran Jayasuriya	323
Saqlain Mushtaq	288
Separamadu Lasith Malinga	318
Shahid Afridi	395
Shane K Warne	293
Shaun M Pollock	393
Waqar Younis	416
Warnakulasuriya Patabendige Ushantha Joseph Chaminda Vaas	400
Wasim Akram	502
Zaheer Khan	282
```



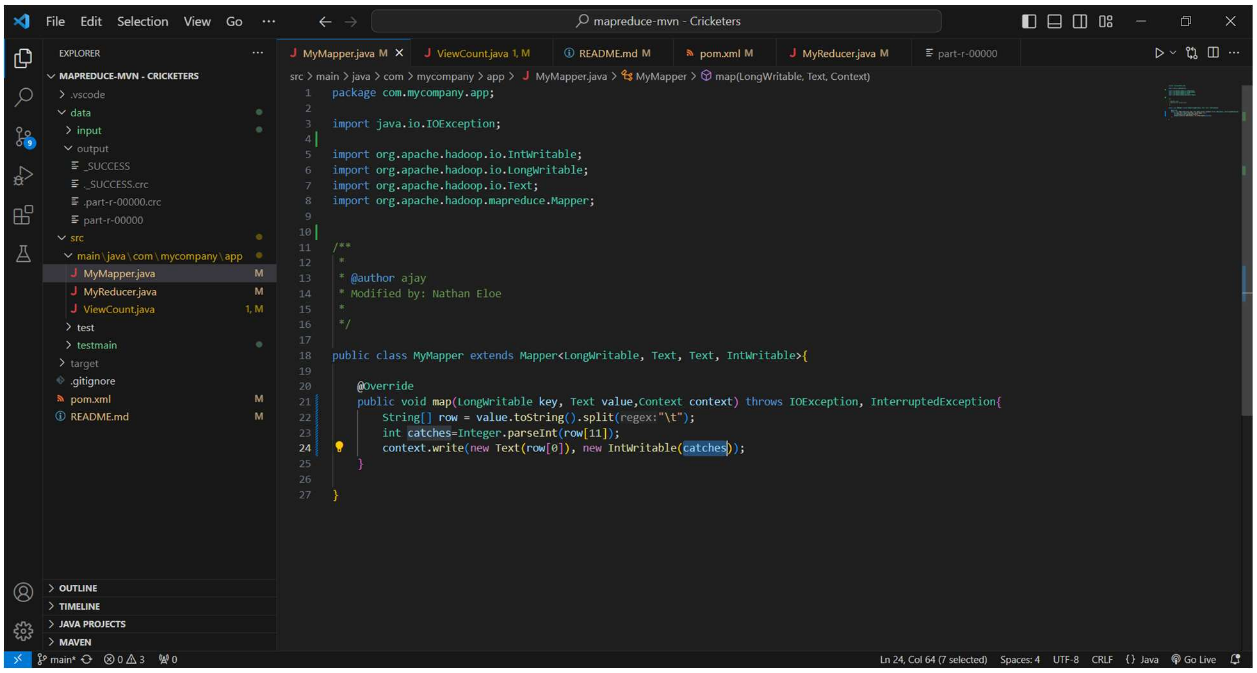
Story:

In the world of cricket, a remarkable group of bowlers has achieved the impressive feat of taking more than 250 wickets each. Legends like Muttiah Muralitharan and Wasim Akram top the list with 534 and 502 wickets, showcasing their unparalleled skill.

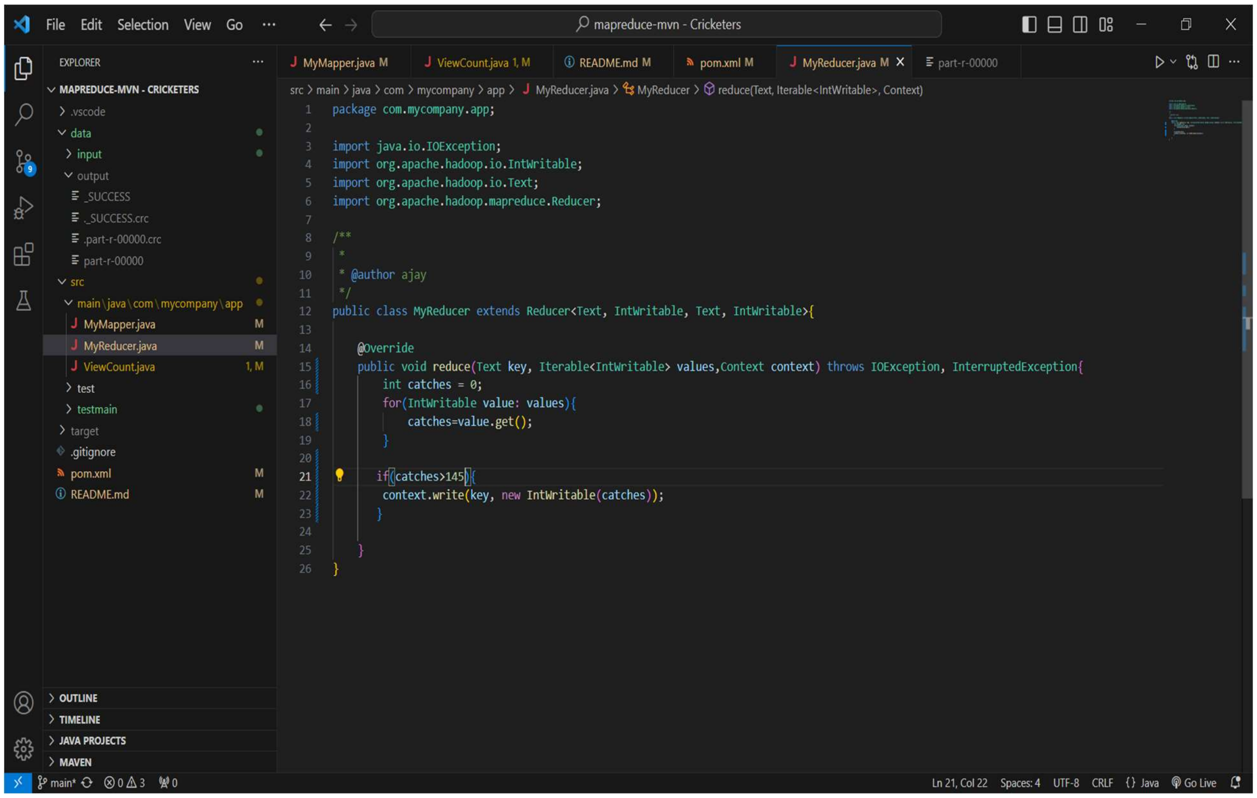
Pace bowler Brett Lee and spinner Anil Kumble stand out with 380 and 337 wickets, respectively. Other notable contributors include Glenn McGrath, Shahid Afridi, Shane Warne, and

Jacques Kallis, each leaving an enduring mark on the game. Together, these bowlers represent a collective legacy of wicket-taking excellence in the rich history of cricket.

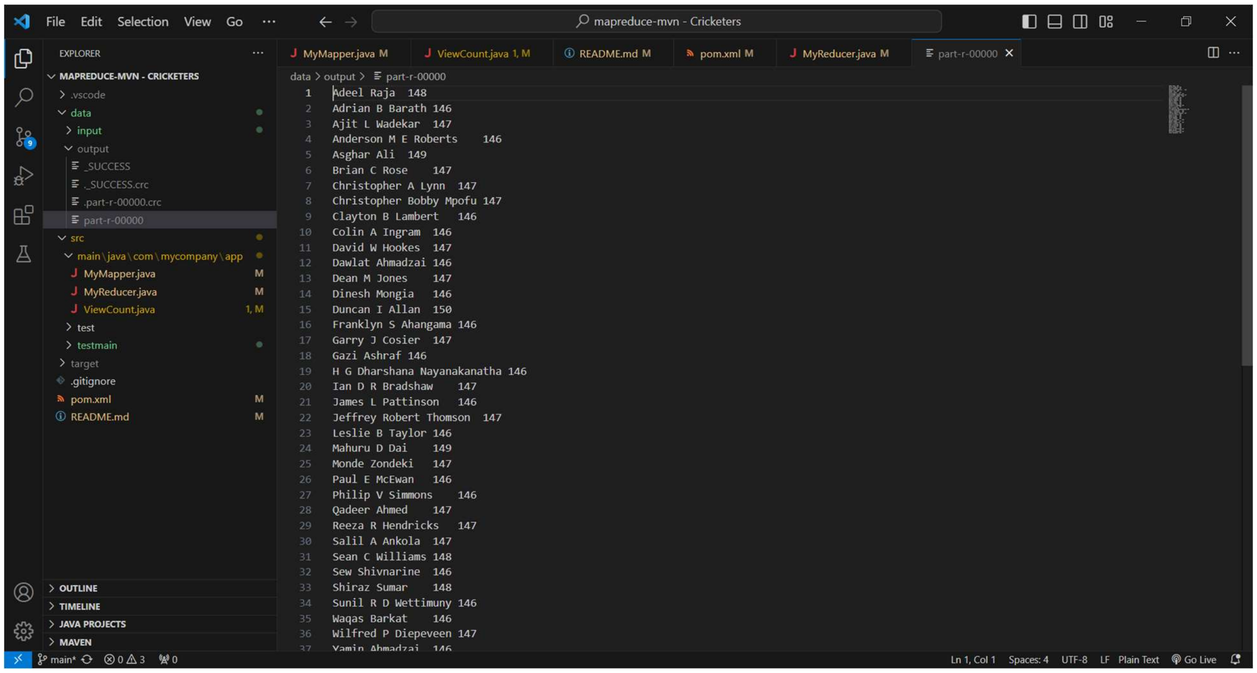
5. **Goal 5: To visualize all the players whose took more than 145 catches.**



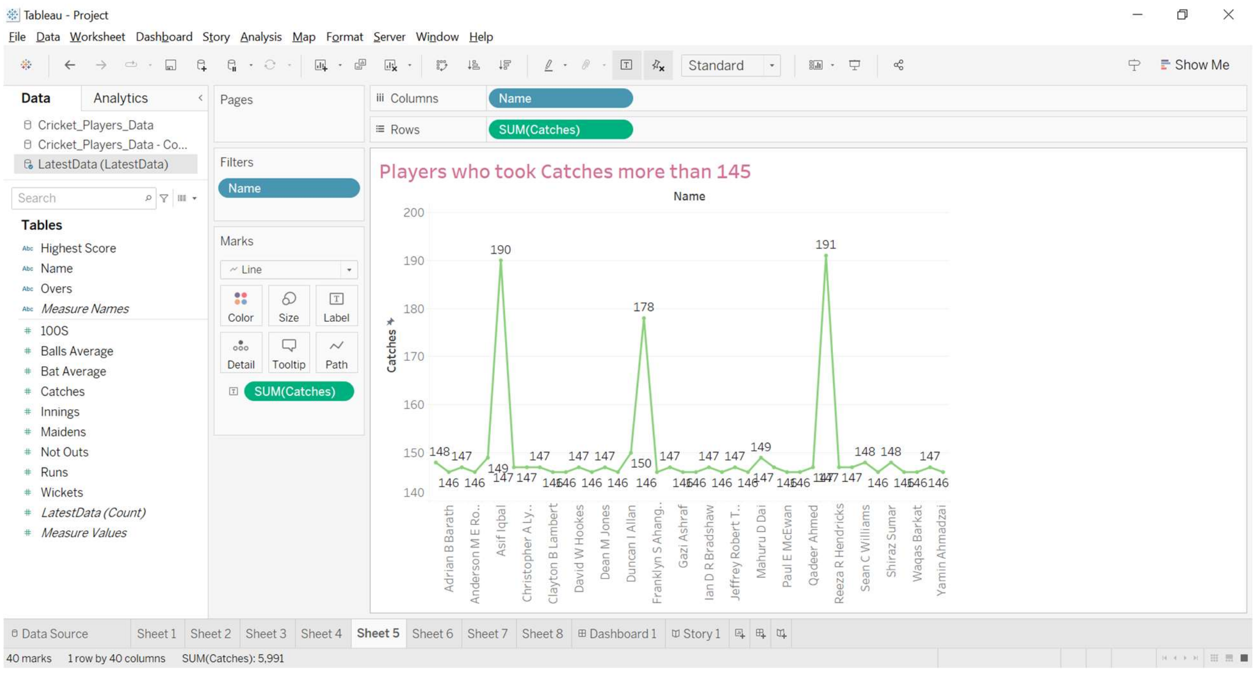
```java
package com.mycompany.app;

import java.io.IOException;

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


/**
 *
 * @author ajay
 * Modified by: Nathan Eloe
 *
 */

public class MyMapper extends Mapper<LongWritable, Text, Text, IntWritable>{

    @Override
    public void map(LongWritable key, Text value,Context context) throws IOException, InterruptedException{
        String[] row = value.toString().split(regex:"\t");
        int catches=Integer.parseInt(row[11]);
        context.write(new Text(row[0]), new IntWritable(catches));
    }

}
```



```java
package com.mycompany.app;

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

/**
 *
 * @author ajay
 */
public class MyReducer extends Reducer<Text, IntWritable, Text, IntWritable>{

    @Override
    public void reduce(Text key, Iterable<IntWritable> values,Context context) throws IOException, InterruptedException{
        int catches = 0;
        for(IntWritable value: values){
            catches=value.get();
        }

       if(catches>145){
        context.write(key, new IntWritable(catches));
       }

    }
}
```



```
Adeel Raja  148
Adrian B Barath 146
Ajit L Wadekar  147
Anderson M E Roberts    146
Asghar Ali  149
Brian C Rose    147
Christopher A Lynn  147
Christopher Bobby Mpofu 147
Clayton B Lambert   146
Colin A Ingram  146
David W Hookes  147
Dawlat Ahmadzai 146
Dean M Jones    147
Dinesh Mongia   146
Duncan I Allan  150
Franklyn S Ahangama 146
Garry J Cosier  147
Gazi Ashraf 146
H G Dharshana Nayanakanatha 146
Ian D R Bradshaw    147
James L Pattinson   146
Jeffrey Robert Thomson  147
Leslie B Taylor 146
Mahuru D Dai    149
Monde Zondeki   147
Paul E McEwan   146
Philip V Simmons    146
Qadeer Ahmed    147
Reeza R Hendricks   147
Salil A Ankola  147
Sean C Williams 148
Sew Shivnarine  146
Shiraz Sumar    148
Sunil R D Wettimuny 146
Waqas Barkat    146
Wilfred P Diepeveen 147
Yamin Ahmadzai  146
```

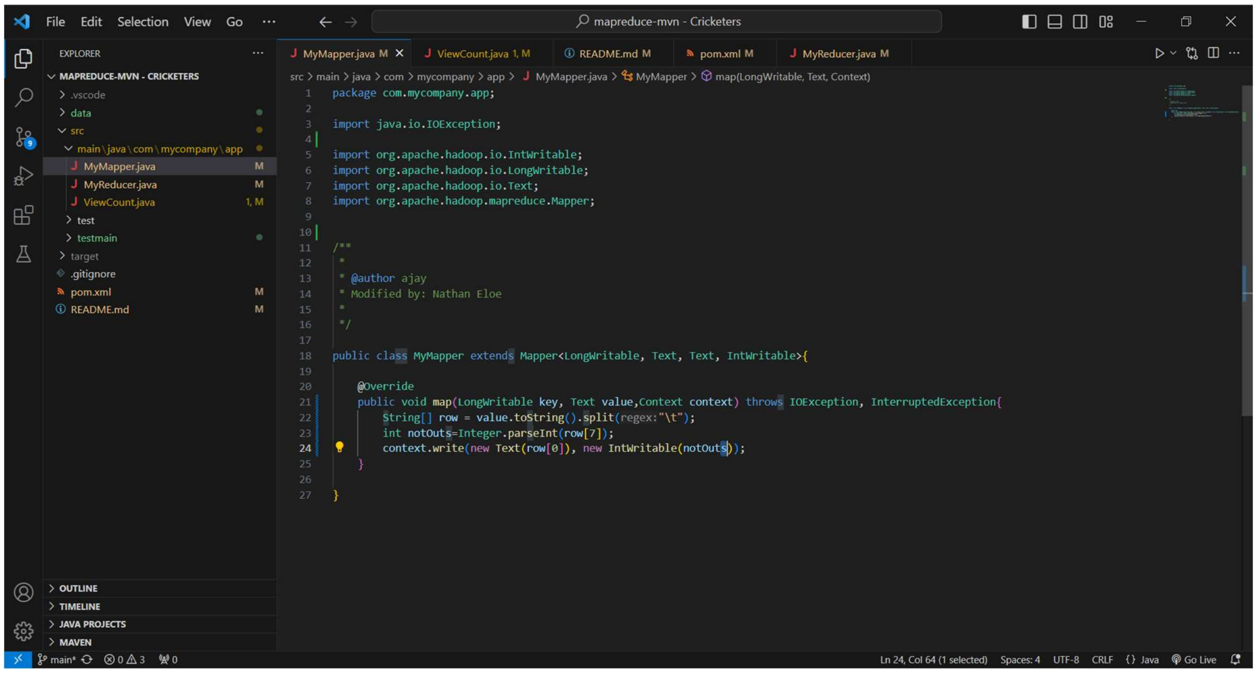


Story:

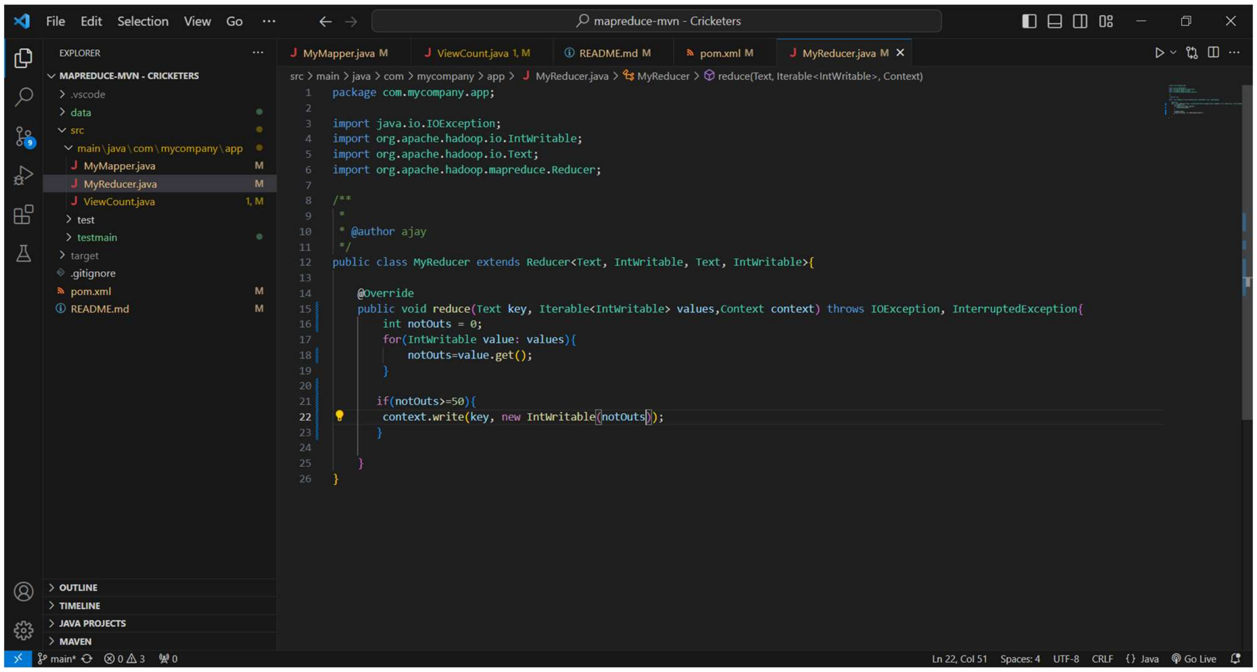
In cricket, there's a select group of players renowned for their exceptional fielding skills, having taken more than 145 catches each. Among them, Adeel Raja and Adrian B Barath lead with 148 and 146 catches, respectively, showcasing their prowess in securing crucial catches on the field. Notable contributors include Ajit L Wadekar, Christopher Lynn, and Duncan I Allan, each with more than 145 catches, highlighting their consistent excellence in fielding.

This group of players, known for their safe hands and agility, has made a significant impact on the defensive aspect of the game, adding to the rich tapestry of cricketing history.

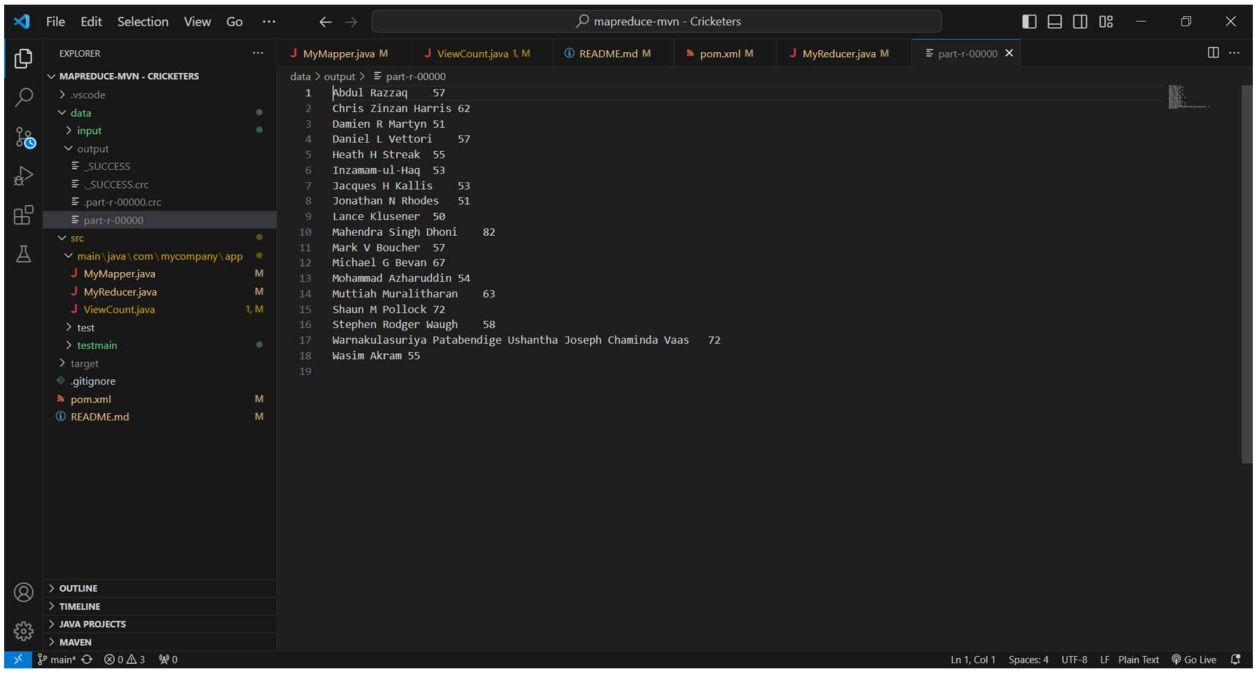
6. **Goal 6: The goal is to visualize all the players who remained not out in more than 50 matches.**


File Edit Selection View Go
mapreduce-mvn - Cricketers
EXPLORER
MAPREDUCE-MVN - CRICKETERS
MyMapper.java M
ViewCount.java 1, M
README.md M
pom.xml M
MyReducer.java M
src > main > java > com > mycompany > app > MyMapper.java > MyMapper > map(LongWritable, Text, Context)
OUTLINE
TIMELINE
JAVA PROJECTS
MAVEN
Ln 24, Col 64 (1 selected) Spaces: 4 UTF-8 CRLF Java Go Live

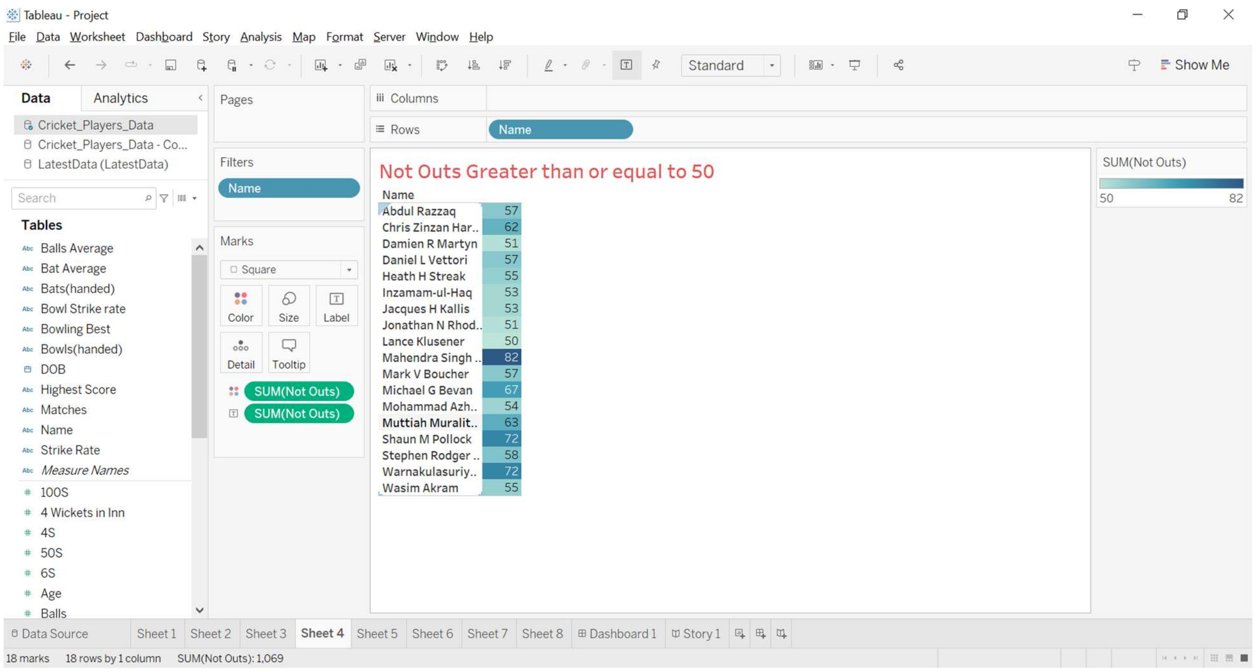
```java
package com.mycompany.app;

import java.io.IOException;

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


/**
 *
 * @author ajay
 * Modified by: Nathan Eloe
 *
 */

public class MyMapper extends Mapper<LongWritable, Text, Text, IntWritable>{

    @Override
    public void map(LongWritable key, Text value,Context context) throws IOException, InterruptedException{
        String[] row = value.toString().split(regex:"\t");
        int notOuts=Integer.parseInt(row[7]);
        context.write(new Text(row[0]), new IntWritable(notOuts));
    }

}
```


File Edit Selection View Go
mapreduce-mvn - Cricketers
EXPLORER
MAPREDUCE-MVN - CRICKETERS
MyMapper.java M
ViewCount.java 1, M
README.md M
pom.xml M
MyReducer.java M
src > main > java > com > mycompany > app > MyReducer.java > MyReducer > reduce(Text, Iterable<IntWritable>, Context)
OUTLINE
TIMELINE
JAVA PROJECTS
MAVEN
Ln 22, Col 51 Spaces: 4 UTF-8 CRLF Java Go Live

```java
package com.mycompany.app;

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

/**
 *
 * @author ajay
 */
public class MyReducer extends Reducer<Text, IntWritable, Text, IntWritable>{

    @Override
    public void reduce(Text key, Iterable<IntWritable> values,Context context) throws IOException, InterruptedException{
        int notOuts = 0;
        for(IntWritable value: values){
            notOuts=value.get();
        }

       if(notOuts>=50){
        context.write(key, new IntWritable(notOuts));
       }

    }
}
```



```
Abdul Razzaq	57
Chris Zinzan Harris 62
Damien R Martyn 51
Daniel L Vettori	57
Heath H Streak	55
Inzamam-ul-Haq	53
Jacques H Kallis	53
Jonathan N Rhodes	51
Lance Klusener	50
Mahendra Singh Dhoni	82
Mark V Boucher	57
Michael G Bevan 67
Mohammad Azharuddin 54
Muttiah Muralitharan	63
Shaun M Pollock 72
Stephen Rodger Waugh	58
Warnakulasuriya Patabendige Ushantha Joseph Chaminda Vaas	72
Wasim Akram 55
```



Not Outs Greater than or equal to 50

Name	
Abdul Razzaq	57
Chris Zinzan Har..	62
Damien R Martyn	51
Daniel L Vettori	57
Heath H Streak	55
Inzamam-ul-Haq	53
Jacques H Kallis	53
Jonathan N Rhod..	51
Lance Klusener	50
Mahendra Singh ..	82
Mark V Boucher	57
Michael G Bevan	67
Mohammad Azh..	54
Muttiah Muralit..	63
Shaun M Pollock	72
Stephen Rodger ..	58
Warnakulasuriy..	72
Wasim Akram	55

SUM(Not Outs) 50 – 82

18 marks 18 rows by 1 column SUM(Not Outs): 1,069

Story:

In cricket, there's a special group of players known for their skill in staying not out in more than 50 matches. The top player in this category is Mahendra Singh Dhoni, who achieved this feat an impressive 82 times, showcasing his ability to finish matches without getting out.

Other notable players in this group include Shaun Pollock and Chaminda Vaas, both with 72 not-out instances. Michael Bevan and Wasim Akram also join this exclusive list, each having more than 50 not-out innings. These players have shown remarkable resilience and contributed significantly to their teams over the years.

Conclusion:

Cricket, with its rich tapestry of players and their multifaceted skills, unfolds a narrative of extraordinary achievements across the realms of batting, bowling, and fielding. The convergence of exceptional talents is evident in the likes of Virat Kohli, Joseph Edward Root, and Rohit G Sharma, whose prowess extends beyond traditional roles, boasting both batting and bowling averages exceeding 40. These versatile players showcase the evolving nature of the sport, where all-round abilities are increasingly valued, redefining the standards of excellence in the cricketing arena.

The bowling domain features a select group of masters, led by Shaun M Pollock, Glenn D McGrath, Kapil Dev, and Muttiah Muralitharan, whose control and strategic brilliance are exemplified by their mastery over 150 maiden overs. Legends such as Courtney A Walsh and Richard John Hadlee, with 185 maiden overs, and Warnakulasuriya Patabendige Ushantha Joseph Chaminda Vaas and Wasim Akram, boasting figures of 277 and 236 maiden overs, contribute to this elite circle. Their enduring excellence and consistent performances have left an indelible imprint on the cricketing landscape, establishing a legacy of skill and precision.

In the realm of century-making excellence, cricket witnesses a stellar cast of players who have etched their names in history by scoring more than 20 centuries. Virat Kohli's remarkable 39 centuries lead the way, standing shoulder to shoulder with legends such as Sachin Tendulkar with 49 centuries. The dynamic duo of Abraham B de Villiers and Christopher H Gayle, with 25 centuries each, adds to the glory, while Ricky Ponting, Sanath Jayasuriya, Kumar Sangakkara, Herschelle Gibbs, Rohit Sharma, Sourav Ganguly, and Tillakaratne Dilshan contribute to this elite circle. Together, these players embody a legacy of century-making excellence, weaving an intricate narrative in the cricketing tapestry.

In the realm of wicket-taking excellence, a formidable group of bowlers has left an indelible mark on the game by surpassing 250 wickets each. Legends like Muttiah Muralitharan and Wasim Akram top the list with 534 and 502 wickets, showcasing unparalleled skill. Brett Lee's pace and Anil Kumble's spin stand out with 380 and 337 wickets, respectively. Contributions from Glenn McGrath, Shahid Afridi, Shane Warne, and Jacques Kallis complete this collective legacy of wicket-taking prowess. Together, these bowlers represent a formidable force that has shaped the dynamic and competitive nature of cricket over the years.

Fielding, an integral aspect of cricket, witnesses a special group of players renowned for their exceptional skills in taking more than 145 catches each. Adeel Raja and Adrian B Barath lead with 148 and 146 catches, showcasing their prowess in securing crucial moments on the field. Ajit L Wadekar, Christopher Lynn, and Duncan I Allan, with more than 145 catches each, underscore their consistent excellence in fielding. This group of players, known for their safe hands and agility, has made a significant impact on the defensive aspect of the game, adding to the rich tapestry of cricketing history.

In a category highlighting resilience and consistency, a group of players has earned recognition for staying not out in more than 50 matches. Mahendra Singh Dhoni, with an impressive 82 instances, leads this exclusive list, demonstrating his ability to finish matches without getting out. Shaun Pollock and Chaminda Vaas, with 72 not-out instances, along with Michael Bevan and Wasim Akram, each boasting more than 50 not-out innings, showcase remarkable resilience. These players, through their tenacity and ability to navigate pressure situations, have made enduring contributions to their teams, defining the essence of cricket as a game of skill, strategy, and unwavering determination.

References:

1. Lewis, A. J. ”Towards fairer measures of player performance in one-day cricket.” Journal of the Operational Research Society 56.7 (2005): 804-815.
2. 2. Beaudoin, David, and Tim B. Swartz. ”The best batsmen and bowlers in one-day cricket.” South African Statistical Journal 37.2 (2003): 203.
3. S. Muthuswamy and S. S. Lam, "Bowler Performance Prediction for One-day International Cricket Using Neural Networks," in Industrial Engineering Research Conference, 2008.
4. I. P. Wickramasinghe, "Predicting the performance of batsmen in test cricket," Journal of Human Sport & Excercise, vol. 9, no. 4, pp. 744-751, May 2014.
5. Kaluarachchi, Amal, and S. Varde Aparna. ”CricAI: A classification based tool to predict the outcome in ODI cricket.” 2010 Fifth International Conference on Information and Automation for Sustainability. IEEE, 2010.
6. Barr, G. D. I., C. G. Holdsworth, and B. S. Kantor. ”Evaluating performances at the 2007 cricket world cup.” South African Statistical Journal 42.2 (2008): 125.
7. Lemmer, Hermanus H. ”The combined bowling rate as a measure of bowling performance in cricket.” South African Journal for Research in Sport, Physical Education and Recreation 24.2 (2002): 37-44.
8. M. G. Jhanwar and V. Pudi, "Predicting the Outcome of ODI Cricket Matches: A Team Composition Based Approach," in European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD 2016 2016), 2016.
9. S. Mukherjee, "Quantifying individual performance in Cricket - A network analysis of batsmen and bowlers," Physica A: Statistical Mechanics and its Applications, vol. 393, pp. 624-637, 2014.
10. C. D. Prakash, C. Patvardhan and C. V. Lakshmi, "Data Analytics based Deep Mayo Predictor for IPL-9," International Journal of Computer Applications, vol. 152, no. 6, pp. 6-10, October 2016.