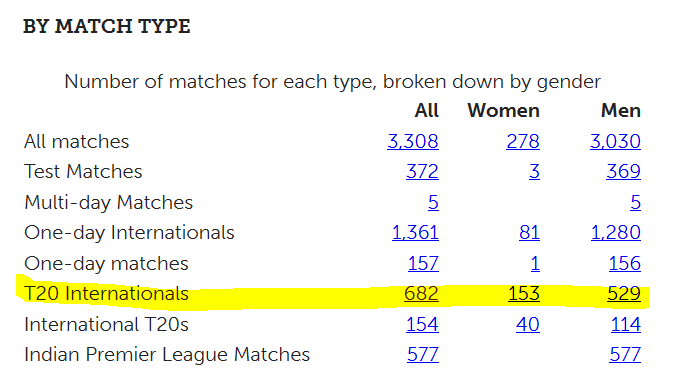
****

Design and analysing Cricket T20 database

**Summary:** My goal in this project is to design a cricket database and analyze the data for finding the key winning factors for a team. In this project, i have used the data files from the website <http://cricsheet.org/downloads/> and we have concentrated on the T20 matches.



**Tools Used:** Excel, R Studio, SQL Server Management Studio.

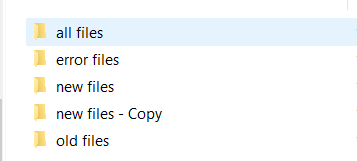
**Project Link:** <https://drive.google.com/open?id=0BxxgIps-0fZAajE5ZnlveEtCQ00>

**Part 1**

**Data cleaning and transforming**

1. R code will download the YAML (682) files from the [**cricsheet**](http://cricsheet.org/downloads/) website and save them in the ***all files*** folder. And further load the downloaded YAML files into the R studio in a loop; compare with old files and then identify the new files.

Update Old Files folder

Clear All Files folder

New Files - copy

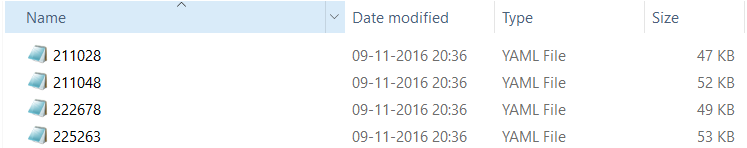
New Files

Compare

All Files

Old Files

**Step 1 - YAML files folder:**



1. Written a file conversion code which will take each file from the ***new files*** folder (shown above) and convert them into equivalent xlsx file; store them in the ***source and source back up*** folder.

Source



New YAML files

Source backup

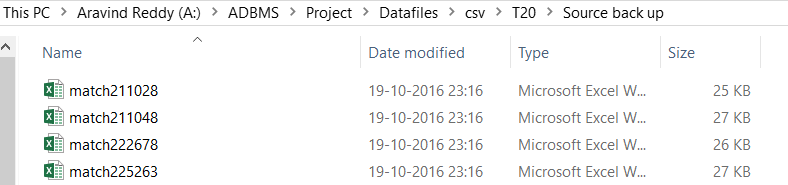
Clear New files folder backup

Clear New files – copy folder

Email error files

Move error files to a error files folder

**Step2** - **YAML equivalent xlsx files:** here R files conversion code will take each of the above yaml file and generate the equivalent xlsx files, if we were not able to convert any yaml, it will be mailed to us and then moved to a separate folder.



1. Now we will take each xlsx file in a loop and then write the R code for Cleaning and Transformation into the meaningful data like match summary, Batting, bowling and partnership summary and export them into respective xlsx files (4 files ).

1.Batting

2.Bowling

3.Partnership

4.Match Summary



Fetch next xlsx file

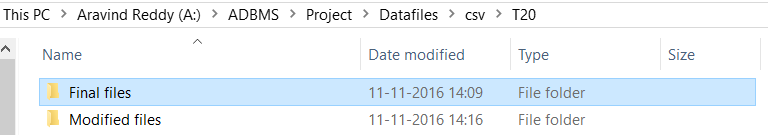
For Loop

All xlsx files

**Step 3** - **Generation of Summary files:** R code will take all the above files in loop and will generate the summary files

Here the ***challenging/problem part*** is: if for suppose we have processed for the 682 files and loaded then into the database, then how to optimally load the data related to the files of the new matches. So we came up with the **solution of storing the summary details of the new match files in a separate location (modified files folder).**

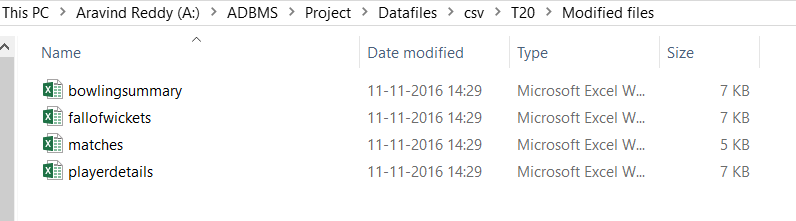
The R code was written in such a way that it will **create new files** in the modified files folder and **update/append the summary details** **of the new files with old files** in the final files folder.



Since we have stored the summary details of the new files in a separate location, we can take only these files and update the respective records in the database instead of working on all the details daily.

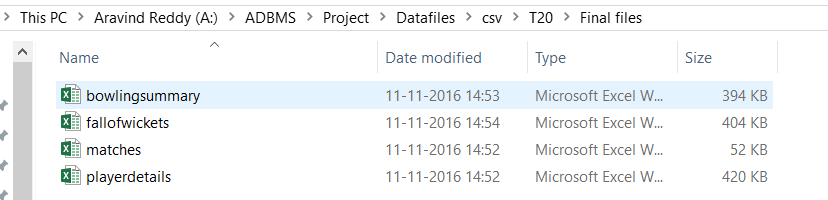
Initially, while running the code for the first time, both the locations: Modified and the Final files folder will have the same files. But in the next run we can observe the differences in the data of the files.

1. Like the modified folder will have the summary details of only the new files



Note the size and the timestamp and the files location

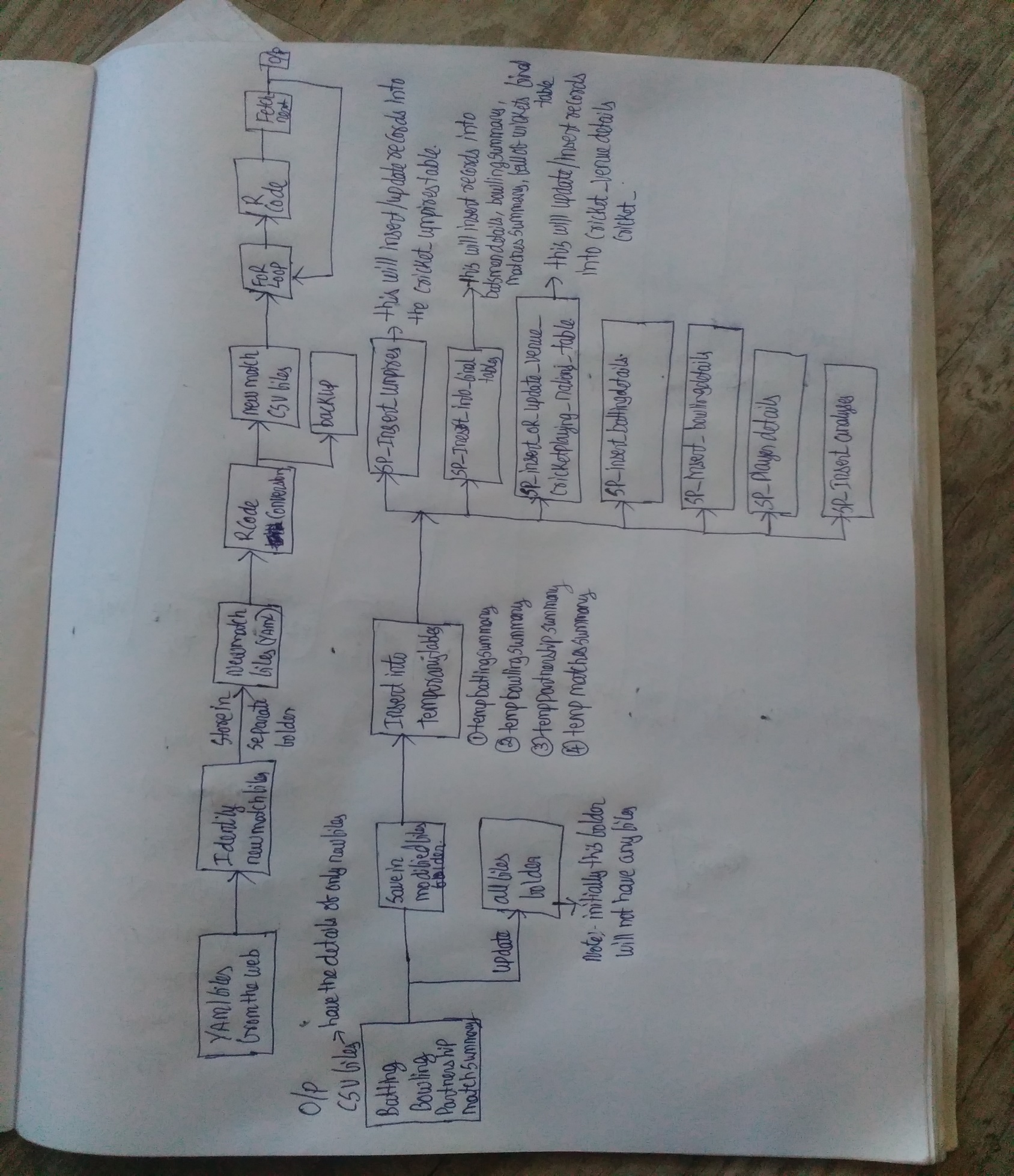
1. Final files folder will have all the details previous matches and the new matches summary details, this is just for our reference.



Note the size and the timestamp and the files location

**Step 4:** Finally loading all these 4 xlsx files from the modified files folder into the SQL database using the stored procedures through SQL Server Management Studio and normalizing the data/ creating appropriate data tables for the analysis.

***Complete data flow architecture diagram:***



**Note:** In the above figure, we have combined SP\_Insert\_batting\_details, SP\_INSERT\_bowlingdetails and SP\_playerdetails into a single stored procedure Insert\_Player\_details.

**Database Design:** SQL script for creating all the tables mentioned below**: **

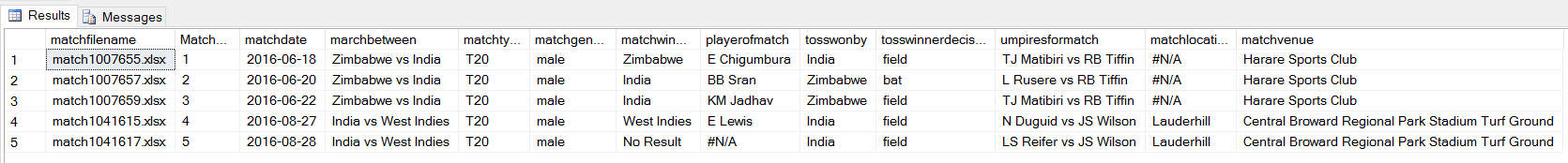
C:\Users\aravi\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Untitled Diagram.png

**Data Loading** into the respective tables:

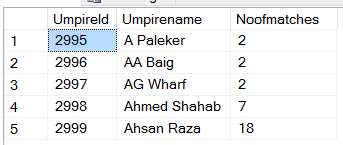
1. Load all the data into the temporary tables mentioned below using the stored procedure: **cricket. INSERT\_into\_Temporarytables**
   * Temporarybatsmendetails
   * Temporarybowlingsummary
   * Temporarymatchsummary
   * Temporaryfallofwickets
2. Using the data from **Temporarymatchsummary** table and the stored procedures **insert\_umpires\_table, Insert\_or\_update\_venue\_cricketplayingnation\_table**; update/insert the data into the below tables:
   * Cricket..umpires
   * Cricket..venuedetails
   * Cricket..cricketplayingnations
3. Using the data from the tables Temporarybatsmendetails, Temporarybowlingsummary and the stored procedure Insert\_Player\_details; update/insert the each batsmen/bowler details in the below mentioned tables.
   * Playerdetails
   * Playerbattingdetails
   * playerbowlingdetails
4. Now we have each player details and the match details, so using the stored procedure load the details into the matchanalysis table which can be further used either in RStudio or Weka for doing the analysis.

Select queries for the tables and also the count of rows in each table: 

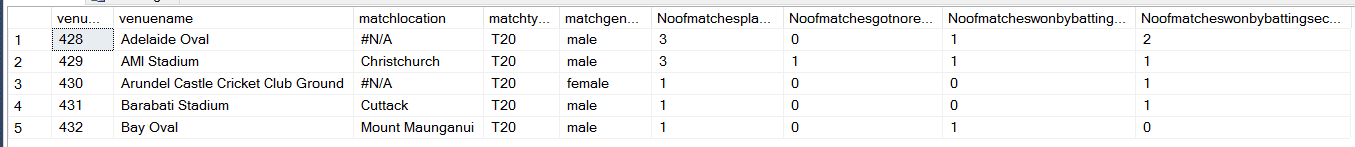
Select top 5 \* from cricket..Matchessummary



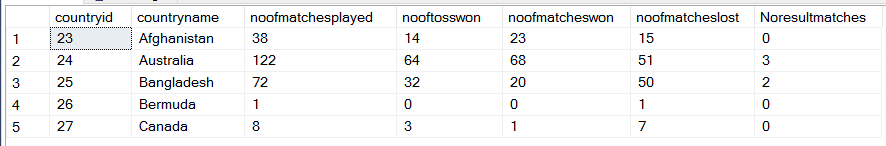
Select top 5 \* from Cricket..umpires



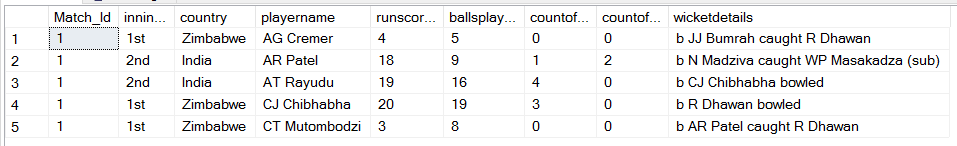
select top 5 \* from Cricket..venuedetails



Select top 5 \* from Cricket..cricketplayingnations



Select top 5 \* from Cricket..batsmendetails



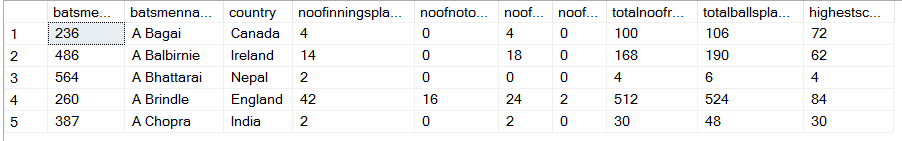
Select top 5 \* from Cricket..bowlingsummary



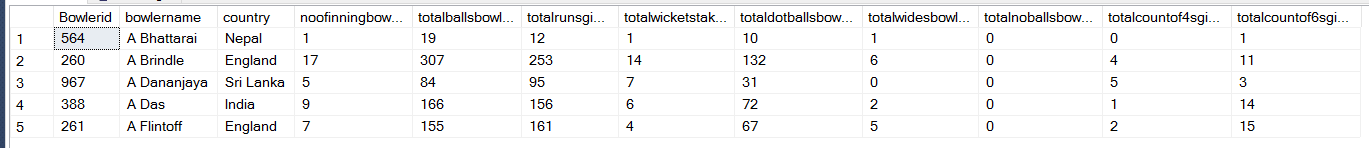
Select top 5 \* from cricket..playerdetails order by country, playername



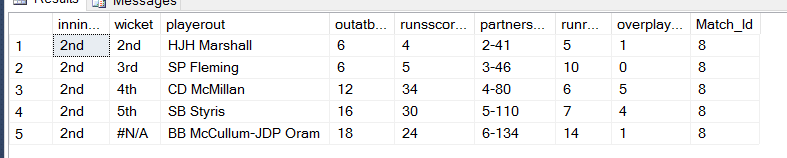
select top 5 \* from cricket..playerbattingdetails



select top 5 \* from cricket..playerbowlingdetails



Select top 5 \* from cricket..fallofwickets



|  |  |
| --- | --- |
| **Table Name** | **Count(\*)** |
| Cricket..Matchessummary | 637 |
| Cricket..umpires | 151 |
| Cricket..venuedetails | 142 |
| Cricket..cricketplayingnations | 22 |
| Cricket..batsmendetails | 10207 |
| Cricket..bowlingsummary | 7593 |
| cricket..playerdetails | 1218 |
| cricket..playerbattingdetails | 1115 |
| cricket..playerbowlingdetails | 817 |
| cricket..fallofwickets | 8963 |
| [Cricket].[dbo].[matchanalysis] | Count will vary based on the team name given in input, since this table will be loaded dynamically |

SELECT [matchbetween] ,[interestedteamname] ,[opposition] ,[match\_winner]

,[matchgender] ,[matchvenue],[tosswonby],[tosswinnerdecision]

,[interestedbattinginnings] ,[playerofmatch],[totalrunscored]

,[Highest\_score],[totalwicketslost] ,[toppartnershipwicket],[toppartnershipruns]

FROM [Cricket].[dbo].[matchanalysis]

**Data Integrity:** Here in the database we have created the

* match\_Id as a primary, and used it as a reference in the bowlingsummary, battingsummary tables.
* Umpire ID as primary key in the Umpire table.
* VenueID as primary key in the venuedetails.
* countryid as primary key in the cricketplayingnations table.
* Playerid as primary key in the playerdetails table which can be used as a refernce in the playerbattingdetails and playerbowlingdetails

**Part 2: Query Writing**

In the 1st part we have explained about the generation of the single Bowling, Batting, Matches, Fallofwickets summary file for the all the T20 matches from yaml files. And the database design. Here we have created total of 6 stored procedures for loading the data into the respective tables mentioned in the database design, explained below:

1. [FINAL\_SINGLE\_POINT\_EXECUTION]

USE [Cricket]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[FINAL\_SINGLE\_POINT\_EXECUTION] Script Date: 19-11-2016 10:24:45 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[FINAL\_SINGLE\_POINT\_EXECUTION]

AS

EXEC INSERT\_into\_Temporarytables

EXEC CRICKET..Insert\_Umpires\_table

EXEC Insert\_OR\_UPDATE\_Venue\_\_Cricketplaying\_nations\_table

EXEC INSERT\_PLAYER\_DETAILS

EXEC INSERT\_into\_Finaltables

GO

Reason: We want to execute all the other 5 stored procedures (highlighted above) for loading into the respective tables in a single go from Rstudio.

1. SP\_Insert\_into\_temporarytables: This stored procedurewill fetch the records from the modified files folder and insert them into the temporary tables.

****

USE [Cricket]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[INSERT\_into\_Temporarytables] Script Date: 19-11-2016 11:00:22 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[INSERT\_into\_Temporarytables]

AS

DELETE FROM Cricket..temporarybatsmendetails

DELETE FROM Cricket..temporarybowlingsummary

DELETE FROM Cricket..temporaryfallofwickets

DELETE FROM Cricket..temporaryMatchessummary

INSERT INTO Cricket..temporaryMatchessummary select \* FROM OPENROWSET('Microsoft.ACE.OLEDB.12.0', 'Excel 12.0;Database=A:\ADBMS\Project\Datafiles\csv\T20\Modified files\matches.xlsx;HDR=YES', 'SELECT \* FROM [Sheet1$]');

INSERT INTO Cricket..temporarybatsmendetails select \* FROM OPENROWSET('Microsoft.ACE.OLEDB.12.0', 'Excel 12.0;Database=A:\ADBMS\Project\Datafiles\csv\T20\Modified files\playerdetails.xlsx;HDR=YES', 'SELECT \* FROM [Sheet1$]')

INSERT INTO Cricket..temporarybowlingsummary select \* FROM OPENROWSET('Microsoft.ACE.OLEDB.12.0', 'Excel 12.0;Database=A:\ADBMS\Project\Datafiles\csv\T20\Modified files\bowlingsummary.xlsx;HDR=YES', 'SELECT \* FROM [Sheet1$]')

INSERT INTO Cricket..temporaryfallofwickets select \* FROM OPENROWSET('Microsoft.ACE.OLEDB.12.0', 'Excel 12.0;Database=A:\ADBMS\Project\Datafiles\csv\T20\Modified files\fallofwickets.xlsx;HDR=YES', 'SELECT \* FROM [Sheet1$]')

GO

1. SP\_Insert\_Umpires\_table: This stored procedure will fetch the records from the table Cricket..temporaryMatchessummary and will first check whether the umpire name is present in the **CRICKET..umpires** table based on the count we will update either the noofmatches played or insert the new umpire details.

****

USE [Cricket]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[Insert\_Umpires\_table] Script Date: 19-11-2016 11:01:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[Insert\_Umpires\_table]

AS

-- FIRST delete all the details from the umpires test table

DELETE FROM CRICKET..umpirestest

DECLARE @delimiter VARCHAR(50)

SET @delimiter=' vs ' -- <=== Here, you can change the delimiter.

;WITH split AS

(

SELECT umpiresformatch, CAST('<M>' + REPLACE([umpiresformatch], @delimiter , '</M><M>') + '</M>' AS XML) AS [umpires]

FROM cricket..temporaryMatchessummary

)

-- INSERT the details of the temporary matchsummary into Umpires test table

INSERT INTO CRICKET..umpirestest

SELECT distinct [umpires].value('/M[1]', 'varchar(50)') FROM split

UNION

SELECT distinct [umpires].value('/M[2]', 'varchar(50)') FROM split

--- declaring the cursor

DECLARE @Umpirename varchar(255)

DECLARE @NoOfMatches int

DECLARE @PREVCNT1 int

DECLARE @PREVCNT2 int

DECLARE db\_Insert\_umpires CURSOR FOR

SELECT umpirename FROM CRICKET..umpirestest

OPEN db\_Insert\_umpires

FETCH NEXT FROM db\_Insert\_umpires INTO @Umpirename

WHILE @@FETCH\_STATUS = 0

BEGIN

set @NoOfMatches = (select count(\*) FROM cricket..temporaryMatchessummary where umpiresformatch like '%' +@Umpirename+ '%' )

set @PREVCNT1= (select count(\*) FROM CRICKET..umpires where Umpirename = @Umpirename )

if (@PREVCNT1 = 0 )

Insert Into CRICKET..umpires (Umpirename,Noofmatches) VALUES(@umpirename, @NoOfMatches)

else

update CRICKET..umpires set Noofmatches = @NoOfMatches + NoOfMatches where Umpirename = @Umpirename

FETCH NEXT FROM db\_Insert\_umpires INTO @Umpirename

END

CLOSE db\_Insert\_umpires;

DEALLOCATE db\_Insert\_umpires;

GO

1. SP\_Insert\_or\_update\_venue\_cricketplayingnations\_table: This stored procedure as the name suggests will update/insert the **venue details and cricketplaying nations** details table shown above.

****

USE [Cricket]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[Insert\_OR\_UPDATE\_Venue\_\_Cricketplaying\_nations\_table] Script Date: 19-11-2016 22:20:06 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[Insert\_OR\_UPDATE\_Venue\_\_Cricketplaying\_nations\_table]

AS

Declare @matchdate varchar(255)

DECLARE @matchlocation varchar(255)

DECLARE @matchvenue varchar(255)

DECLARE @matchtype varchar(10)

DECLARE @matchgender varchar(30)

DECLARE @matchwinner varchar(255)

DECLARE @tosswonby varchar(255)

DECLARE @tosswinnerdecision varchar(255)

DECLARE @team1 varchar(255)

DECLARE @team2 varchar(255)

DECLARE @batting1st varchar(255)

DECLARE @batting2nd varchar(255)

DECLARE @scenario varchar(255)

--- Splitting the team names in the temporary match summary table

DECLARE @delimiter VARCHAR(50)

SET @delimiter=' vs ' -- <=== Here, you can change the delimiter.

;WITH SPlit AS

(

SELECT matchdate,matchlocation, matchvenue,matchtype,matchgender, marchbetween,

CAST('<M>'+ REPLACE(marchbetween, @delimiter,'</M><M>' ) + '</M>' AS XML) AS teams,

tosswonby, tosswinnerdecision,

'null' as batting1st, 'NULL' as batting2nd, matchwinner, 'null' as scenario

FROM cricket..temporaryMatchessummary

)

SELECT matchdate,matchlocation, matchvenue, matchtype,matchgender, marchbetween,

teams.value('/M[1]', 'varchar(50)') as team1,

teams.value('/M[2]', 'varchar(50)') as team2,

tosswonby, tosswinnerdecision,batting1st,batting2nd, matchwinner, scenario into #temp

FROM SPlit

ALTER table #temp ALTER COLUMN batting1st varchar(100)

ALTER table #temp ALTER COLUMN batting2nd varchar(100)

ALTER table #temp ALTER COLUMN scenario varchar(100)

-- Fetching the match winner details like 1st batting or 2nd batting and scenario matches played in venue's

DECLARE db\_split CURSOR FOR

SELECT matchdate,matchlocation, matchvenue, matchtype,matchgender,

team1,team2,tosswonby, tosswinnerdecision, batting1st, batting2nd, matchwinner, scenario

FROM #temp

OPEN db\_split

FETCH NEXT FROM db\_split INTO @matchdate,@matchlocation, @matchvenue, @matchtype, @matchgender, @team1, @team2,

@tosswonby, @tosswinnerdecision,@batting1st, @batting2nd, @matchwinner, @scenario

WHILE @@FETCH\_STATUS = 0

BEGIN

if (@tosswinnerdecision = 'bat')

update #temp set batting1st = @tosswonby where

matchdate = @matchdate and matchvenue = @matchvenue and matchtype = @matchtype and matchgender = @matchgender and team1 = @team1 and team2 = @team2

update #temp set batting2nd = CASE WHEN batting1st = team1 then team2 else team1 END where

matchdate = @matchdate and matchvenue = @matchvenue and matchtype = @matchtype and matchgender = @matchgender and team1 = @team1 and team2 = @team2

if (@tosswinnerdecision = 'field')

update #temp set batting2nd = @tosswonby where

matchdate = @matchdate and matchvenue = @matchvenue and matchtype = @matchtype and matchgender = @matchgender and team1 = @team1 and team2 = @team2

update #temp set batting1st = CASE WHEN batting2nd = team1 then team2 ELSE team1 end where

matchdate = @matchdate and matchvenue = @matchvenue and matchtype = @matchtype and matchgender = @matchgender and team1 = @team1 and team2 = @team2

update #temp set scenario = CASE WHEN matchwinner = batting1st then '1st'

WHEN matchwinner = batting2nd then '2nd'

when matchwinner = 'No Result' then 'NR' END

where matchdate = @matchdate and matchvenue = @matchvenue and matchtype = @matchtype and matchgender = @matchgender and team1 = @team1 and team2 = @team2

FETCH NEXT FROM db\_split INTO @matchdate,@matchlocation, @matchvenue, @matchtype, @matchgender, @team1, @team2,

@tosswonby, @tosswinnerdecision,@batting1st, @batting2nd, @matchwinner, @scenario

END

CLOSE db\_split;

DEALLOCATE db\_split;

--- updating or inserting the details in the CRICKET..venuedetails

DECLARE @totalnofmatchesplayed INT

DECLARE @matcheswonbybatting1stteam INT

DECLARE @matcheswonbybatting2ndteam INT

DECLARE @noresult INT

DECLARE @cntt2 INT

DECLARE db\_Insert\_UPDATE\_VENUE\_TABLE CURSOR FOR

Select t1.matchvenue, t1.matchlocation,

t1.matchtype, t1.matchgender,

count(\*) as totalnofmatchesplayed ,

(select count(\*) from #temp t2 where t2.matchvenue = t1.matchvenue and t2.matchtype = t1.matchtype and t2.matchgender = t1.matchgender and t2.scenario = '1st' )AS matcheswonbybatting1stteam,

(select count(\*) from #temp t2 where matchvenue = t1.matchvenue and matchtype = t1.matchtype and matchgender = t1.matchgender and scenario = '2nd') as matcheswonbybatting2ndteam,

(select count(\*) from #temp t2 where matchvenue = t1.matchvenue and matchtype = t1.matchtype and matchgender = t1.matchgender and scenario = 'NR') as noresult

from #temp t1

group by matchlocation, matchvenue, matchtype,matchgender

OPEN db\_Insert\_UPDATE\_VENUE\_TABLE

FETCH NEXT FROM db\_Insert\_UPDATE\_VENUE\_TABLE INTO @matchvenue, @matchlocation, @matchtype, @matchgender, @totalnofmatchesplayed, @matcheswonbybatting1stteam,@matcheswonbybatting2ndteam,@noresult

WHILE @@FETCH\_STATUS = 0

BEGIN

set @cntt2 = (select count(\*) FROM CRICKET..venuedetails where venuename = @matchvenue and matchlocation = @matchlocation and matchtype = @matchtype and matchgender = @matchgender )

if (@cntt2 = 0 )

Insert Into CRICKET..venuedetails (venuename,matchlocation, matchtype,matchgender, Noofmatchesplayed,Noofmatchesgotnoresult,Noofmatcheswonbybattingfirst, Noofmatcheswonbybattingsecond)

VALUES (@matchvenue, @matchlocation, @matchtype, @matchgender, @totalnofmatchesplayed,@noresult,@matcheswonbybatting1stteam, @matcheswonbybatting2ndteam )

else

update CRICKET..venuedetails set Noofmatchesplayed = Noofmatchesplayed + @totalnofmatchesplayed,

Noofmatchesgotnoresult = Noofmatchesgotnoresult + @noresult ,

Noofmatcheswonbybattingfirst = Noofmatcheswonbybattingfirst + @matcheswonbybatting1stteam,

Noofmatcheswonbybattingsecond = Noofmatcheswonbybattingsecond + @matcheswonbybatting2ndteam

where venuename = @matchvenue and matchlocation = @matchlocation and matchtype = @matchtype and matchgender = @matchgender

FETCH NEXT FROM db\_Insert\_UPDATE\_VENUE\_TABLE INTO @matchvenue, @matchlocation, @matchtype, @matchgender, @totalnofmatchesplayed, @matcheswonbybatting1stteam,@matcheswonbybatting2ndteam,@noresult

END

CLOSE db\_Insert\_UPDATE\_VENUE\_TABLE;

DEALLOCATE db\_Insert\_UPDATE\_VENUE\_TABLE;

-- updating or inserting the details in the Cricket..cricketplayingnations

DECLARE @countryname varchar(255)

DECLARE @noofmatchesplayed INT

DECLARE @noofmatcheswon INT

DECLARE @nooftosswon INT

DECLARE @noofmatcheslost INT

DECLARE @NOOFRESULTmatches INT

DECLARE @cntt3 INT

DECLARE db\_Insert\_UPDATE\_CRICKET\_PLAYING\_NATIONS\_TABLE CURSOR FOR

select distinct team1 from #temp

union

select distinct team2 from #temp

OPEN db\_Insert\_UPDATE\_CRICKET\_PLAYING\_NATIONS\_TABLE

FETCH NEXT FROM db\_Insert\_UPDATE\_CRICKET\_PLAYING\_NATIONS\_TABLE INTO @countryname

WHILE @@FETCH\_STATUS = 0

BEGIN

SET @noofmatchesplayed = ( select count(\*) from #temp where marchbetween like '%'+ @countryname + '%' )

SET @nooftosswon = (select count(\*) from #temp where tosswonby = @countryname)

SET @noofmatcheswon = (select count(\*) from #temp where matchwinner = @countryname)

SET @NOOFRESULTmatches = ( select count(\*) from #temp where marchbetween like '%'+ @countryname + '%' and scenario = 'NR')

SET @noofmatcheslost = @noofmatchesplayed - @noofmatcheswon - @NOOFRESULTmatches

set @cntt3 = (select count(\*) from Cricket..cricketplayingnations where countryname = @countryname)

if (@cntt3 = 0)

INSERT INTO Cricket..cricketplayingnations(countryname,noofmatchesplayed,nooftosswon,noofmatcheswon,noofmatcheslost,Noresultmatches)

VALUES(@countryname,@noofmatchesplayed,@nooftosswon,@noofmatcheswon,@noofmatcheslost,@NOOFRESULTmatches)

else

UPDATE Cricket..cricketplayingnations set noofmatchesplayed = noofmatchesplayed + @noofmatchesplayed,

nooftosswon = nooftosswon + @nooftosswon,

noofmatcheswon = noofmatcheswon + @noofmatcheswon,

noofmatcheslost = noofmatcheslost + @noofmatcheslost,

Noresultmatches = Noresultmatches + @NOOFRESULTmatches

where countryname = @countryname

FETCH NEXT FROM db\_Insert\_UPDATE\_CRICKET\_PLAYING\_NATIONS\_TABLE INTO @countryname

END

CLOSE db\_Insert\_UPDATE\_CRICKET\_PLAYING\_NATIONS\_TABLE;

DEALLOCATE db\_Insert\_UPDATE\_CRICKET\_PLAYING\_NATIONS\_TABLE;

DROP table #temp

GO

1. SP\_INSERT\_PLAYER\_DETAILS: the goal of this stored procedure is to insert/update the details in the below mentioned tables.

* cricket..playerdetails – Insert/Update
* cricket..playerbattingdetails – Insert/Update
* cricket..playerbowlingdetails – Insert/Update

****

1. SP\_INSERT\_into\_Finaltables: This stored procedurewill fetch the records from the temporary tables created and insert them into the final tables mentioned below.

* Cricket..Matchessummary
* Cricket..batsmendetails
* Cricket..bowlingsummary
* Cricket..fallofwicket

USE [Cricket]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[INSERT\_into\_Finaltables] Script Date: 19-11-2016 18:44:37 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[INSERT\_into\_Finaltables]

AS

INSERT INTO Cricket..Matchessummary select \* FROM Cricket..temporaryMatchessummary

INSERT INTO Cricket..batsmendetails select \* FROM Cricket..temporarybatsmendetails

INSERT INTO Cricket..bowlingsummary select \* FROM Cricket..temporarybowlingsummary

INSERT INTO Cricket..fallofwickets select \* FROM Cricket..temporaryfallofwickets

GO

****

**Interesting Query:**

Use CRICKET

declare @team1 varchar(255)

set @team1 = 'India'

select MS.marchbetween, @team1 as interestedteam ,RTRIM(LTRIM(replace( replace(MS.marchbetween,@team1,''),'vs',''))) as opposition,MS.matchwinner, MS.match\_id, MS.matchgender,MS.matchvenue, MS.tosswonby,MS.tosswinnerdecision,

RTRIM(LTRIM(replace( CASE when MS.tosswinnerdecision = 'bat' Then MS.tosswonby ELSE replace(MS.marchbetween,MS.tosswonby,'') END ,'vs',''))) as battingfirst,

MS.playerofmatch,

CASE when MS.matchwinner = (Select distinct country from Cricket..batsmendetails where Match\_Id = MS.match\_id and innings = '1st') Then '1st' Else '2nd' End as battinginningsofmatchwinningteam,

CASE WHEN MS.matchwinner = (Select distinct country from Cricket..batsmendetails where Match\_Id = MS.match\_id and innings = '1st') Then '2nd' ELSE '1st' END as battinginningsofmatchlosingteam,

(select top(1) playername from Cricket..batsmendetails where Match\_Id = MS.Match\_Id and country = @team1 Order by runscored desc, ballsplayed asc) as HSplayername,

(select top(1) runscored from Cricket..batsmendetails where Match\_Id = MS.Match\_Id and country = @team1 Order by runscored desc, ballsplayed asc) as Hscore,

(select count(\*) from Cricket..batsmendetails where Match\_Id = MS.Match\_Id and country = @team1 and wicketdetails != 'Not Out' ) as totalwicketslost,

(select sum(runsscored) from cricket..fallofwickets where Match\_Id = MS.Match\_Id and innings in (select distinct innings from Cricket..batsmendetails where Match\_Id = MS.Match\_Id and country = @team1 )) as totalrunsscored,

(select sum(runsscored) from cricket..fallofwickets where Match\_Id = MS.Match\_Id and innings in (select distinct innings from Cricket..batsmendetails where Match\_Id = MS.Match\_Id and country = @team1 ) and wicket in ('1st','2nd','3rd','4th','5th')) as Toorderpartnershipruns,

(select top(1) wicket from cricket..fallofwickets where Match\_Id = MS.Match\_Id and innings in (select distinct innings from Cricket..batsmendetails where Match\_Id = MS.Match\_Id and country = @team1 ) order by runsscored desc) as Toppartnershipwicket,

(select max(runsscored) from cricket..fallofwickets where Match\_Id = MS.Match\_Id and innings in (select distinct innings from Cricket..batsmendetails where Match\_Id = MS.Match\_Id and country = @team1 ) ) as Toppartnershipruns

from Matchessummary MS

where marchbetween like '%'+@team1+'%'

and matchwinner != 'No Result'

order by match\_id



We have used the tables:

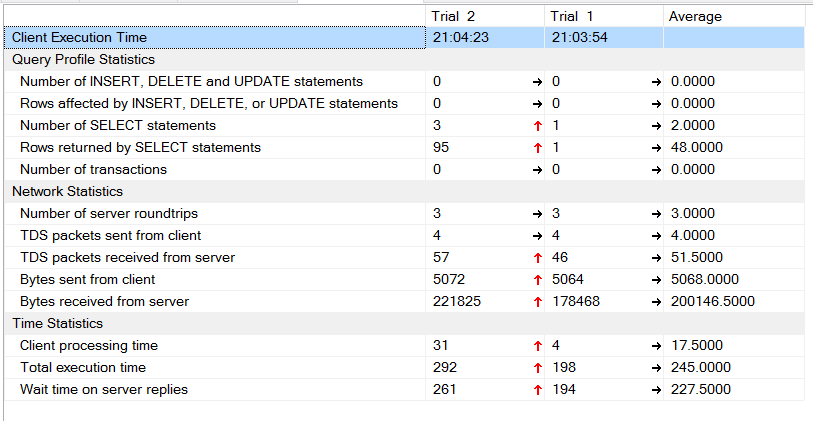
* cricket..fallofwickets
* Cricket..batsmendetails
* Cricket.. Matchessummary

This query output could be used for match analysis, we have made the query a bit complex. We can get the same output in R studio in a simple way, but we just want to learn on how to use select statements inside a select statement, how to use case when in the select statement.

**Purpose:** The above query takes a country name as input and will give the details for all the matches played by that country

* Venue
* Matchwinner
* Tosswonby
* Tosswinnerdecision
* Matchwinnerbattinginnings
* Playerofmatch
* Highestscore
* Highestscoreby
* Totalrunsscored
* Totalwicketslost
* Toporderpartnership
* Highestpartnershipwicket
* toppartnesrshipofgiventeam

Client statistics:



Using all the above details we will run a decision tree and will check What are key winnings factors for a particular team.

**Part 3: Performance Tuning**

In this section, you should highlight any experiments run as part of the project related to performance tuning.  Experiments with different indexing strategies, optimizer changes, transaction isolation levels, function-based indexes, and table partitioning can all be interesting. Remember to look at different types of queries (e.g., point, range, scan), execution plans, and I/O burden.  For each experiment include the following: (1) purpose of the experiment, (2) steps followed to run the experiment, (3) key results (include screenshots, figures, and/or tables to help highlight results), and (4) a discussion of the results that explains what happened and why.

**Reason** for the creation of the below 3 indexes: For fetching the match analysis we often query the below table on the columns like match\_id, country, innings

Created Index on the table Cricket..batsmendetails on the columns match\_id and country

create index IX\_cricket\_batsmendetails on Cricket..batsmendetails(match\_id,playername)

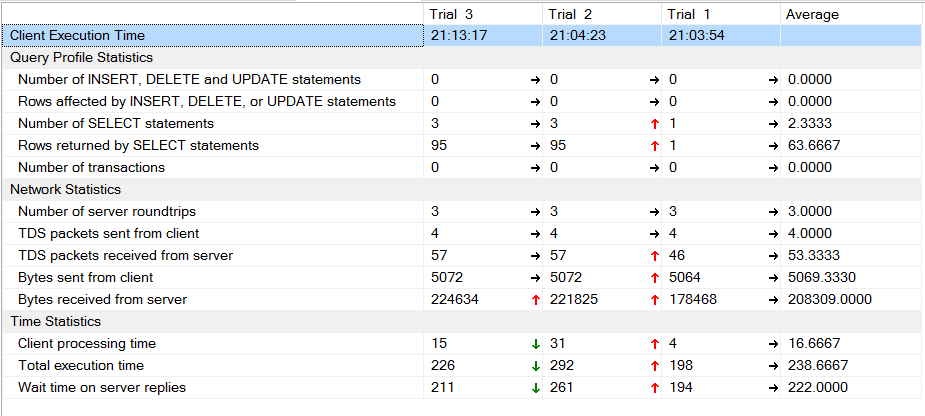
Created index on the table Cricket..batsmendetails on the columns match\_id and innings

create index IX\_cricket\_batsmendetails\_matchid\_innings on Cricket..batsmendetails(match\_id,innings)

Created index on the table cricket..fallofwickets on the column match\_id and innings

create index IX\_cricket\_fall\_of\_wickets on cricket..fallofwickets(match\_id,innings)

The client statistics of the above interesting query after creating the above 3 indexes:



However since we are having less no.of rows, so we are not able to show significant difference in the statistics. But creation of index will definitely show difference when there are more number of rows.

**Reason** for the creation of the below index: While updating the records in the table cricket..playerdetails, we would check whether the player name is there for the available for the country or not. With the creation of the index, we can fetch the row count quickly and can either insert/update records in the table.

create index IX\_cricket\_playerdetails on cricket..playerdetails(country,playername)

**Reason** for the creation of the below index: While updating the records in the table cricket..playerbattingdetails, we would check whether the batsmen is new or already played for the country before or not. Accordingly we will either insert /update records in the table. With the creation of the index, we can fetch the row count quickly and can either insert/update records in the table.

create index IX\_cricket\_player\_batting\_details on cricket..playerbattingdetails(batsmenid,country)

**Reason** for the creation of the below index: While updating the records in the table cricket.. playerbowlingdetails, we would check whether the bowler is new or already played for the country before or not. Accordingly we will either insert /update records in the table. With the creation of the index, we can fetch the row count quickly and can either insert/update records in the table.

create index IX\_cricket\_player\_bowling\_details on cricket..playerbowlingdetails (bowlerid,country)

The above 3 indexes are more important, because all the tables insertion/updation is done by a single stored procedure through cursor operation, so if we fetch the row count quickly for each player, then the insertion will be done easily and also for the updation also this index will be helpful.

Part 4: **Data Mining**

**Scenario:** Suppose tomorrow there is a match between India vs New Zealand, and my manager has asked about what are the key factors for both the teams to win the match.

Here we have created a table valued function: analysis, which we will take team name/country as input and will return table as output.



USE CRICKET

GO

create function analysis(@team1 varchar(255))

RETURNS @analysis Table(matchbetween varchar(255), result varchar(10),interestedteamname varchar(255),opposition varchar(255), matchwinner varchar(255), matchgender varchar(255), matchvenue varchar(255),

tosswonby varchar(255), tosswinnerdecision varchar(255),interestedteambattinginnings varchar(10), playerofmatch varchar(255), totalrunsscored int, highestscore int, totalwicketlost int, toporderpartnership int, toppartnershipwicket varchar(10), toppartnershipruns int)

As

BEGIN

Insert into @analysis

select MS.marchbetween,

CASE WHEN MS.matchwinner = @team1 Then 'won' Else 'lost' end as result,

@team1 as interestedteam ,

RTRIM(LTRIM(replace( replace(MS.marchbetween,@team1,''),'vs',''))) as opposition,

MS.matchwinner, MS.matchgender,MS.matchvenue, MS.tosswonby,MS.tosswinnerdecision,

CASE WHEN RTRIM(LTRIM(replace( CASE when MS.tosswinnerdecision = 'bat' Then MS.tosswonby ELSE replace(MS.marchbetween,MS.tosswonby,'') END ,'vs',''))) = @team1 THEN '1st' ELSE '2nd' END as interestedteambattinginnings,

MS.playerofmatch,

(select sum(runsscored) from cricket..fallofwickets where Match\_Id = MS.Match\_Id and innings in (select distinct innings from Cricket..batsmendetails where Match\_Id = MS.Match\_Id and country = @team1 )) as totalrunsscored,

(select top(1) runscored from Cricket..batsmendetails where Match\_Id = MS.Match\_Id and country = @team1 Order by runscored desc, ballsplayed asc) as Hscore,

(select count(\*) from Cricket..batsmendetails where Match\_Id = MS.Match\_Id and country = @team1 and wicketdetails != 'Not Out' ) as totalwicketslost,

(select sum(runsscored) from cricket..fallofwickets where Match\_Id = MS.Match\_Id and innings in (select distinct innings from Cricket..batsmendetails where Match\_Id = MS.Match\_Id and country = @team1 ) and wicket in ('1st','2nd','3rd','4th','5th')) as Toorderpartnershipruns,

(select top(1) wicket from cricket..fallofwickets where Match\_Id = MS.Match\_Id and innings in (select distinct innings from Cricket..batsmendetails where Match\_Id = MS.Match\_Id and country = @team1 ) order by runsscored desc) as Toppartnershipwicket,

(select max(runsscored) from cricket..fallofwickets where Match\_Id = MS.Match\_Id and innings in (select distinct innings from Cricket..batsmendetails where Match\_Id = MS.Match\_Id and country = @team1 ) ) as Toppartnershipruns

from cricket..Matchessummary MS

where marchbetween like '%'+@team1+'%'

and matchwinner != 'No Result'

order by match\_id

RETURN;

END;

Since the function is defined in the SQL database; executing the function and then loading the data into the csv/xlsx file and then finally loading it into R studio for analysis will take a lot of time.

* So we have connected SQL to R studio
* then executed the function and saved the output table into a data frame, on which we can perform decision tree analysis
* Further mail the graphs to our personnel mail id, like reporting the results.

# connecting SQL server to R

odbcChannel <- odbcConnect("Cricket")

# executed the function related to team New Zealand

analysisNZ <- sqlQuery(odbcChannel,"SELECT \* FROM analysis('New Zealand')")

analysisNZ1 <- analysisNZ[,c(-1,-4,-5,-6,-11)]

# Save the plot in the below location.

jpeg(file = "A:/ADBMS/Project/analysis/newzealand.jpeg")

# Create the tree.

output.tree <- ctree( result~., data = analysisNZ1)

# Plot the tree.

plot(output.tree)

# Save the file.

dev.off()

# executed the function related to team India

analysisInd <- sqlQuery(odbcChannel,"SELECT \* FROM analysis('India')")

analysisIndt1 <- analysisInd[,c(-1,-4,-5,-6,-11)]

# Save the plot in the below location

jpeg(file = "A:/ADBMS/Project/analysis/india.jpeg")

# Create the tree.

output.tree <- ctree( result~., data = analysisIndt1)

# Plot the tree.

plot(output.tree)

# Save the file.

dev.off()

### send email with attachments code

send.mail(from = "aravindreddy727090@gmail.com",

to = c( "Mkrishnamallik@gmail.com"),

subject = "India vs New Zealand match analysis",

body = "please find the attachments for the analysis details ",

smtp = list(host.name = "aspmx.l.google.com", port = 25),

authenticate = FALSE,

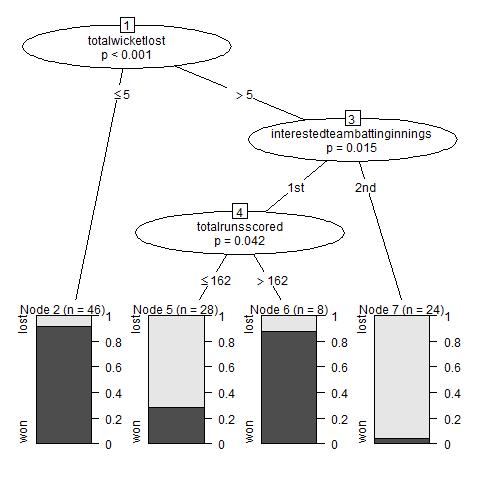
send = TRUE,

attach.files = c("A:/ADBMS/Project/analysis/india.jpeg", "A:/ADBMS/Project/analysis/newzealand.jpeg"),

debug = TRUE)

**output graphs:**

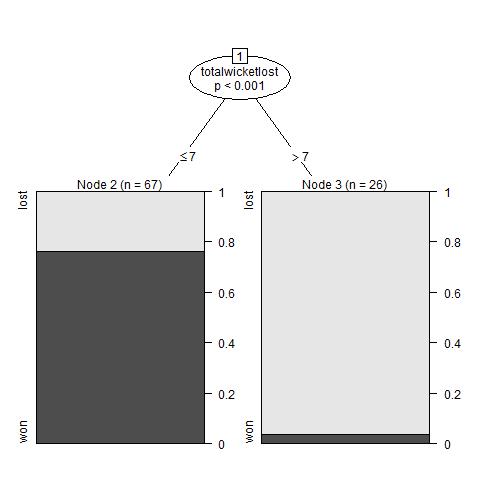
**Team: New Zealand**



**Explanation:**

* For all the previous matches played by New Zealand, they have won more matches if the totalwicketlost is less than 5.
* And they have won less no of matches if they have batted in the 2nd innings
* And they have won more no of matches if they have batted 1st and have scored more than 162 runs.

**Team: India**



**Explanation:**

* India has won more no of matches if they have lost less than 7 wickets in a match (count = 47).
* The win percentage is very less if India has lost more than 7 wickets in a match.