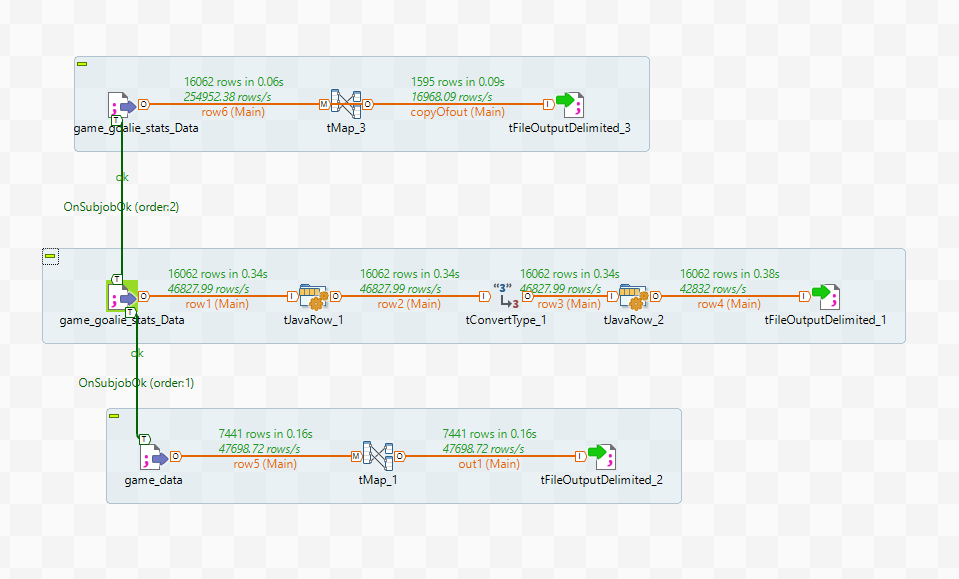
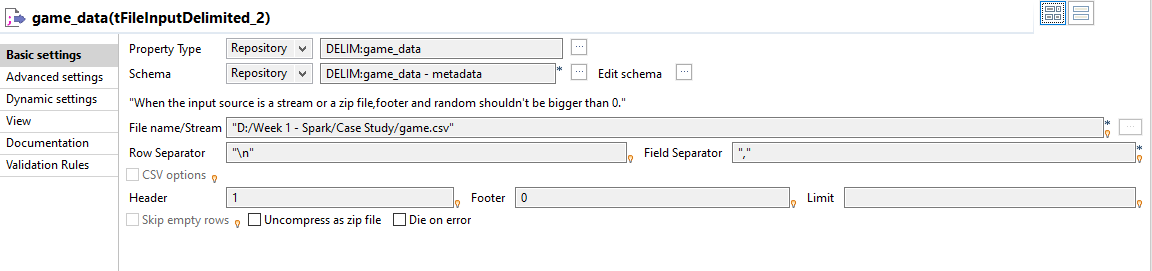
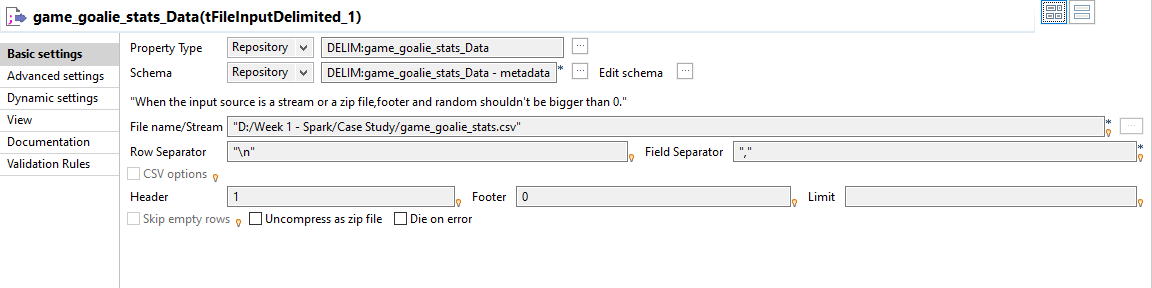
**Case study on Games Exercise**

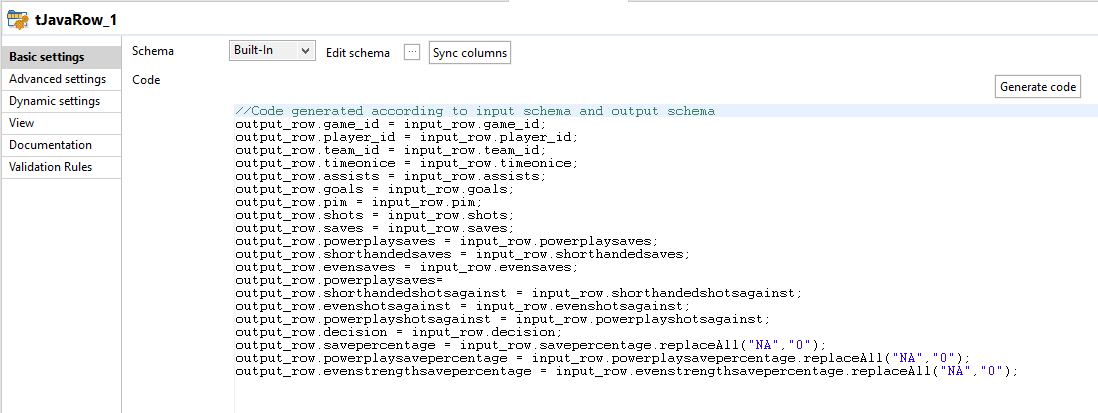
**Quality checks and profiling(Using Talend)**

**3) A)** Finding null values and replace them

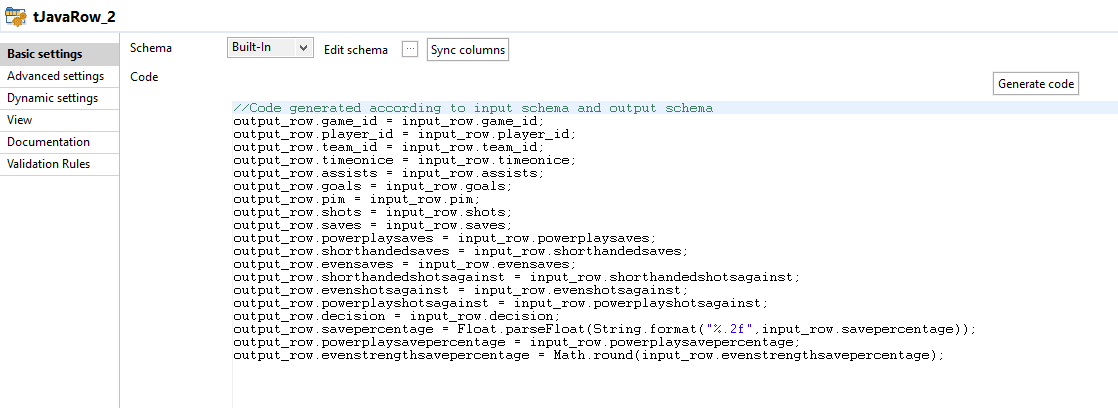
****

****

****

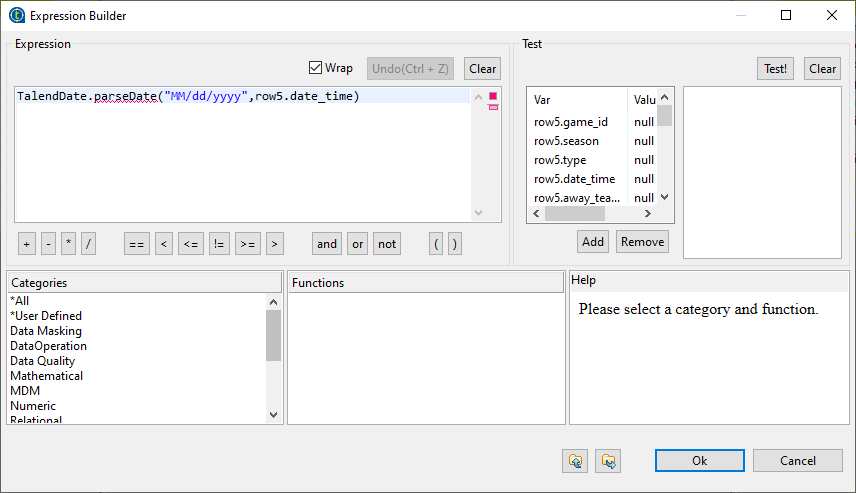
****

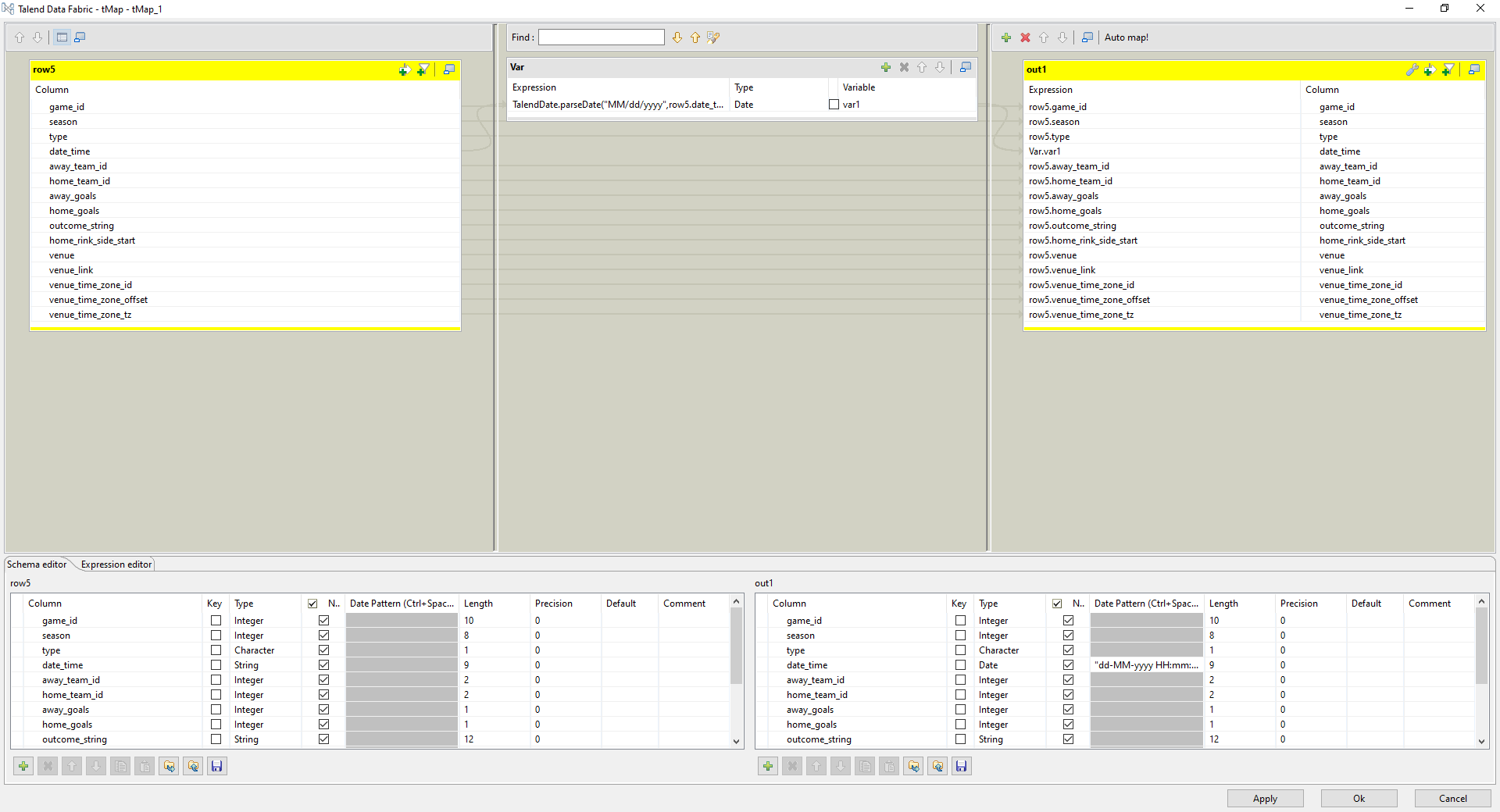
**3) b)** Round off decimal values up to 2 places

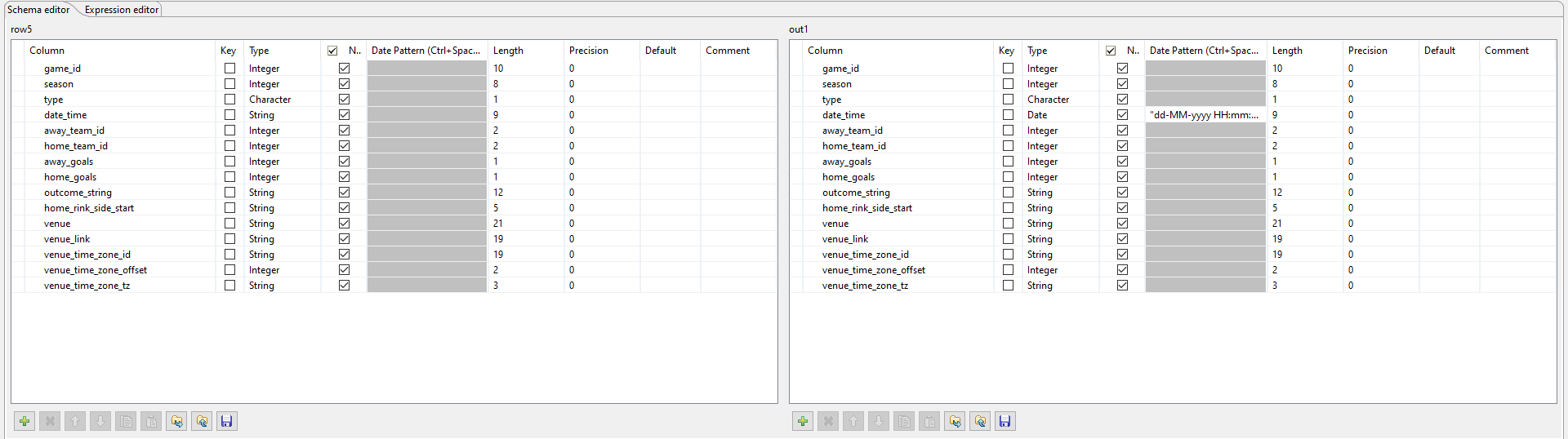
****

**4) Standardization (Date, Number)**

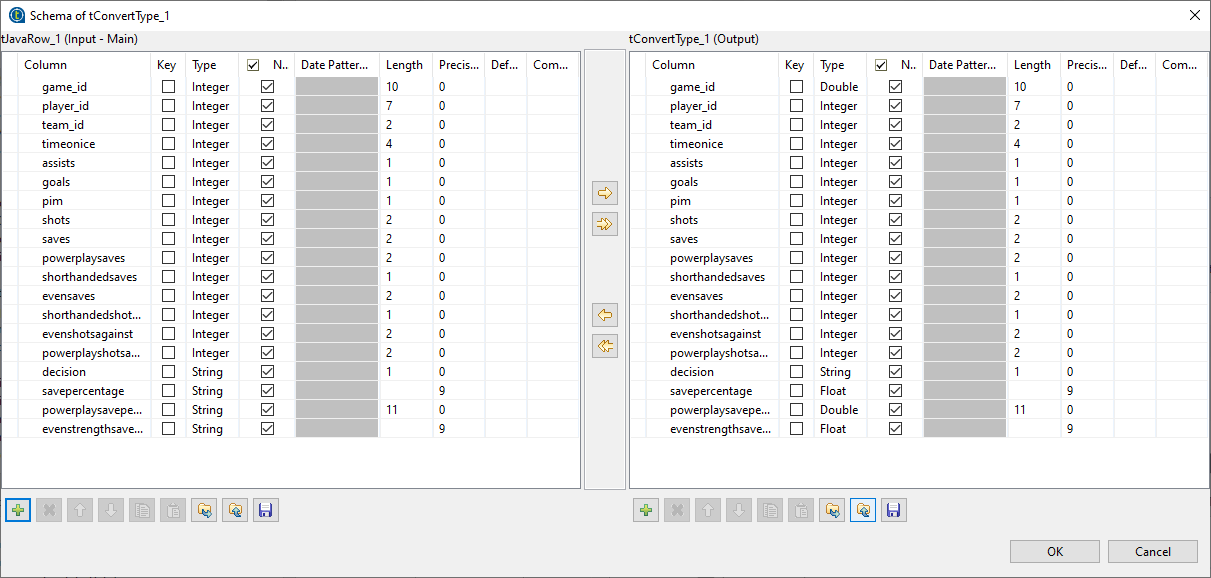
**4) A)** Convert dates to “dd-mm-yyyy hh:mm:ss” format

****

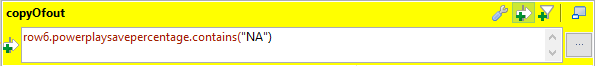
****

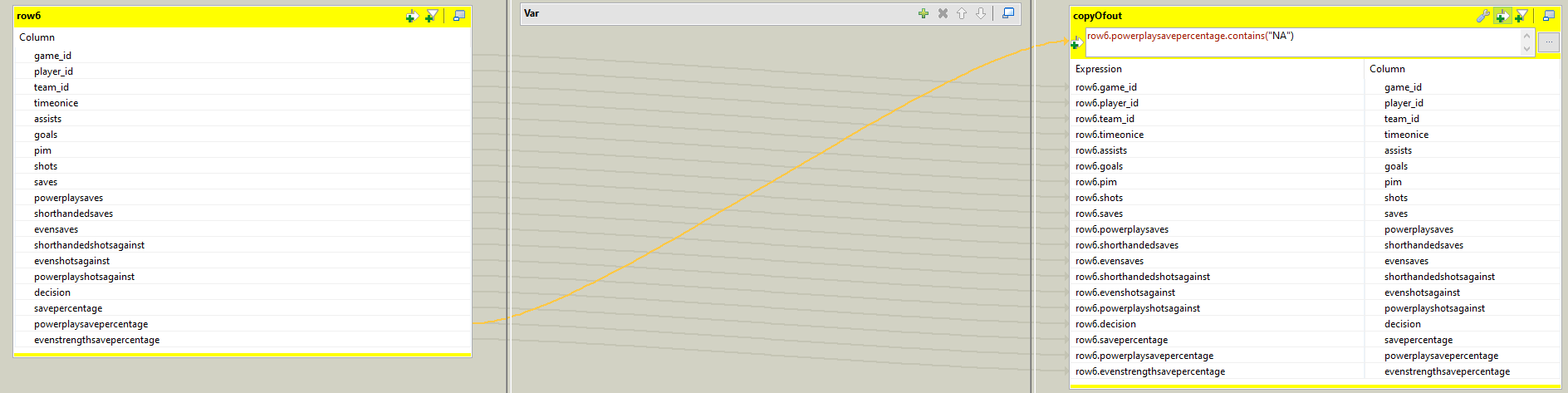
****

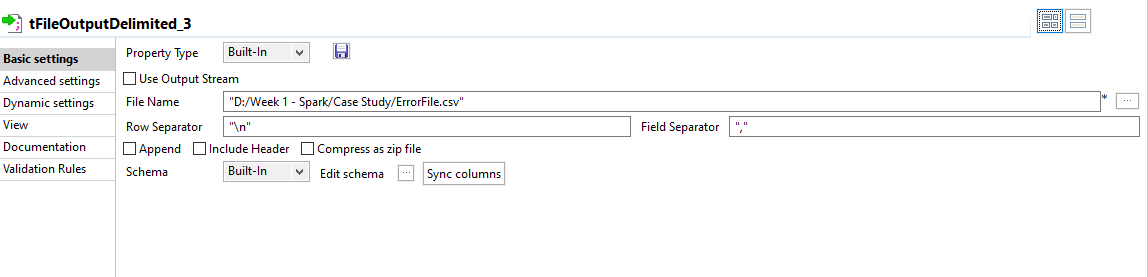
**4) B**) Convert float to integer

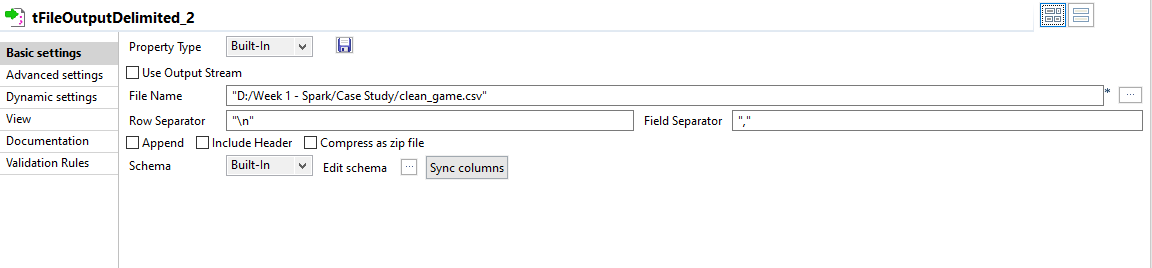
****

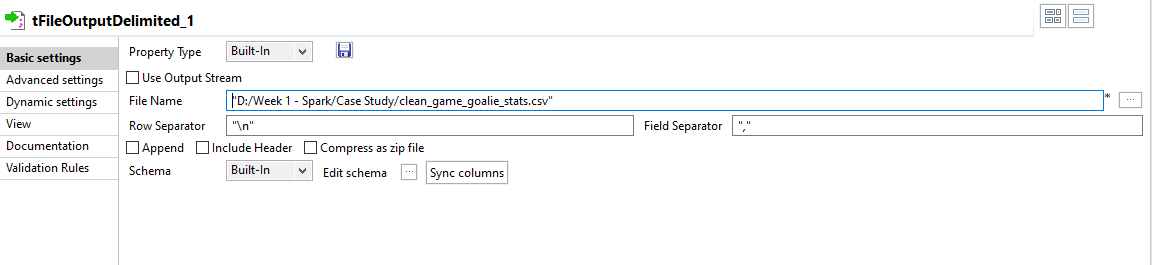
**Sending the null value records to separate file**

****

****

****

****

****

**Using spark**

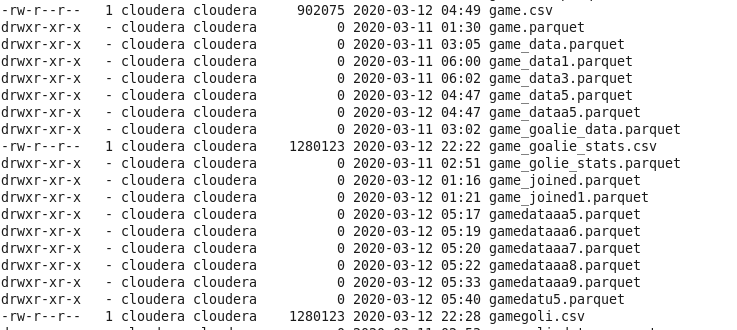
**3) Quality checks and profiling (HiveQL)**

**Copying from local to hdfs (uncleaned datasets)**

hadoop fs -copyFromLocal "/home/user/Desktop/game.csv"

hadoop fs -copyFromLocal "/home/user/Desktop/gamegolie.csv"

Hadoop fs -ls



**Creating dataframe in spark**

**gamegoli.csv**

1. val gameGoali1 = sc.textFile("/user/cloudera/gamegoli.csv")
2. case class gamegolieDat1(game\_id: Long,player\_id: Long,team\_id: Integer,timeonice: Long,assists: Integer,goals: Integer,pim: Integer,shots: Integer,saves: Integer,powerplaysaves: Integer,shorthandedsaves: Integer,evensaves: Integer,shorthandedshotsagainst: Integer,evenshotsagainst: Integer,powerplayshotsagainst: Integer,decision: String,savepercentage:String,powerplaysavepercentage: String,evenstrengthsavepercentage: String)
3. val mapGameGolieData5 = gameGoali1.map(\_.split(",")).map(g => gamegolieDat1(g(0).toLong, g(1).toLong, g(2).toInt, g(3).toLong, g(4).toInt,g(5).toInt,g(6).toInt,g(7).toInt,g(8).toInt,g(9).toInt,g(10).toInt,g(11).toInt,g(12).toInt,g(13).toInt,g(14).toInt,g(15),g(16),g(17),g(18)))
4. val gameshiftsDataReadyy1 = mapGameGolieData5.toDF()
5. gameshiftsDataReadyy1.write.parquet("/user/cloudera/parquetfile3.parquet")
6. val gameDataRea = sqlContext.read.parquet("/user/cloudera/parquetfile3.parquet")

**game.csv**

1. val gameFile1 = sc.textFile("/user/cloudera/game.csv")
2. case class gameData3(game\_id: Long, season: Long, game\_type: String, date\_time: String,away\_team\_id: Integer,home\_team\_id: Integer,away\_goals: Integer,home\_goals: Integer,outcome: String,home\_rink\_side\_start: String,venue: String,venue\_link: String, venue\_time\_zone\_id: String, venue\_time\_zone\_offset: Integer,venue\_time\_zone\_tz: String)
3. val mapGameData = gameFile1.map(\_.split(",")).map(g => gameData3(g(0).toLong,g(1).toLong,g(2),g(3),g(4).toInt,g(5).toInt,g(6).toInt,g(7).toInt,g(8),g(9),g(10),g(11),g(12),g(13).toInt,g(14)))
4. val gameDataReadyy = mapGameData.toDF()
5. gameDataReadyy.write.parquet("/user/cloudera/gameparquet12.parquet")
6. val gameDataReadyy1 = sqlContext.read.parquet("/user/cloudera/gameparquet12.parquet")

**Quality checks and profiling(Using Talend)**

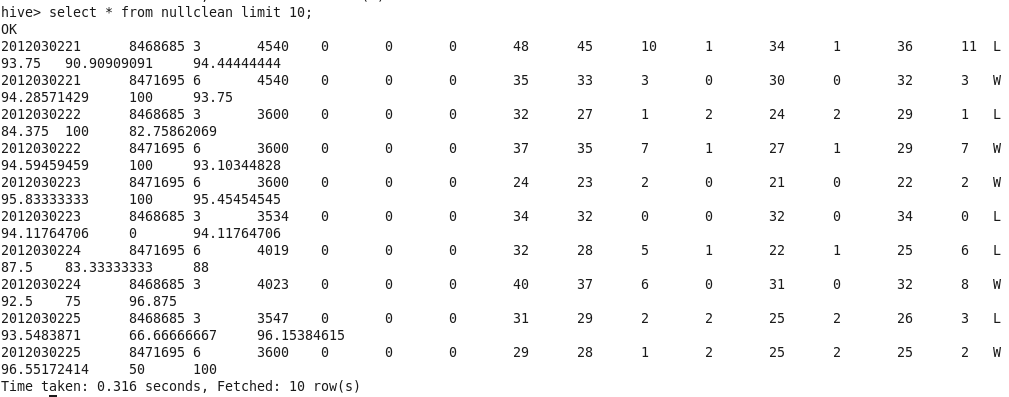
**3) A) Finding null values and replace them**

val updatedDf = gameshiftsDataReadyy1.withColumn("powerplaysavepercentage", regexp\_replace(col("powerplaysavepercentage"), "NA", "0"))

Loading to hive tables

updatedDf.registerTempTable("nullclean")

sqlContext.sql("create table casestudy.nullclean as select \* from nullclean")

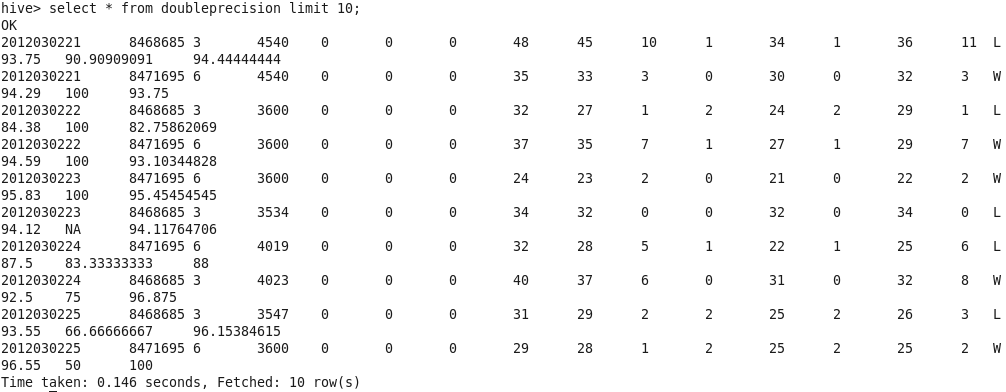


**3) b)Round off decimal values up to 2 places**

val upd= gameshiftsDataReadyy1.withColumn("savepercentage",round(gameshiftsDataReadyy1.col("savepercentage"),2))

upd.registerTempTable("doubleprecision")

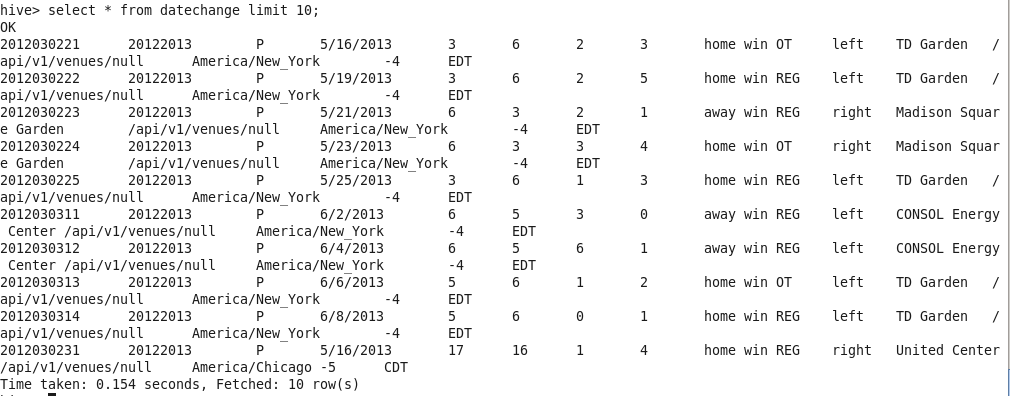
sqlContext.sql("create table casestudy.doubleprecision as select \* from doubleprecision")



**4)Standardization (Date, Number)**

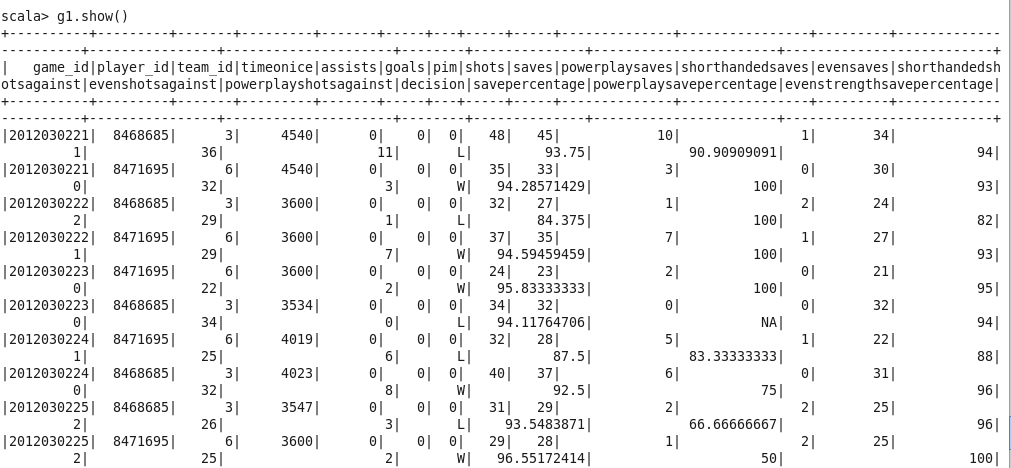
**4) A) Convert dates to “dd-mm-yyyy hh:mm:ss” format**

* val modify = gameDataReadyy.withColumn("date\_time",date\_format(unix\_timestamp($"date\_time","MM/dd/yyyy").cast("timestamp"),"dd-MM-yyyy HH:mm:ss"))
* gameDataReadyy.registerTempTable("datechange")
* sqlContext.sql("create table casestudy.datachange as select \* from datachange"



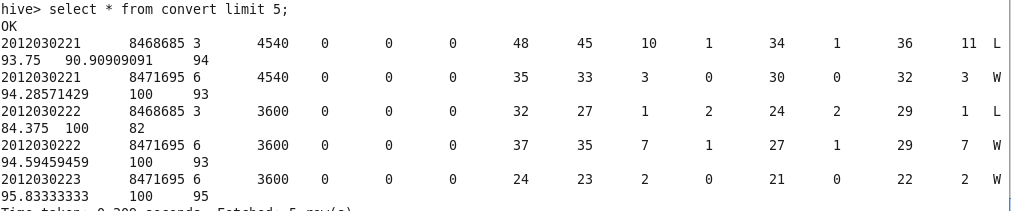
4)B)**Convert float to Integer**

val g1=gameDataRea.withColumn("evenstrengthsavepercentage",$"evenstrengthsavepercentage".cast("Int"))



g1.registerTempTable("convert")

sqlContext.sql("create table casestudy.convert as select \* from convert")



**5)** **Flattening-3 De-normalized tables (Data frames)**

**Copying from local to hdfs (cleaned datasets)**

clean\_game.csv

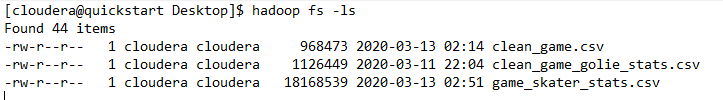
hadoop fs -copyFromLocal "/home/user/Desktop/clean\_game.csv”

clean\_game\_golie\_stats.csv

hadoop fs -copyFromLocal "/home/user/Desktop/clean\_game\_golie\_stats.csv”

game\_skater\_stats.csv

hadoop fs -copyFromLocal "/home/user/Desktop/game\_skater\_stats.csv



**Creating dataframe in spark and creating temp table in spark**

1) clean\_game.csv

i) val gameFile = sc.textFile("/user/cloudera/game.csv")

ii) case class gameData(game\_id: Long, season: Long, game\_type: String, date\_time: String,away\_team\_id: Integer,home\_team\_id: Integer,away\_goals: Integer,home\_goals: Integer,outcome: String,home\_rink\_side\_start: String,venue: String,venue\_link: String, venue\_time\_zone\_id: String, venue\_time\_zone\_offset: Integer,venue\_time\_zone\_tz: String)

iii) val mapGameData = gameFile.map(\_.split(",")).map(g => gameData(g(0).toLong,g(1).toLong,g(2),g(3),g(4).toInt,g(5).toInt,g(6).toInt,g(7).toInt,g(8),g(9),g(10),g(11),g(12),g(13).toInt,g(14)))

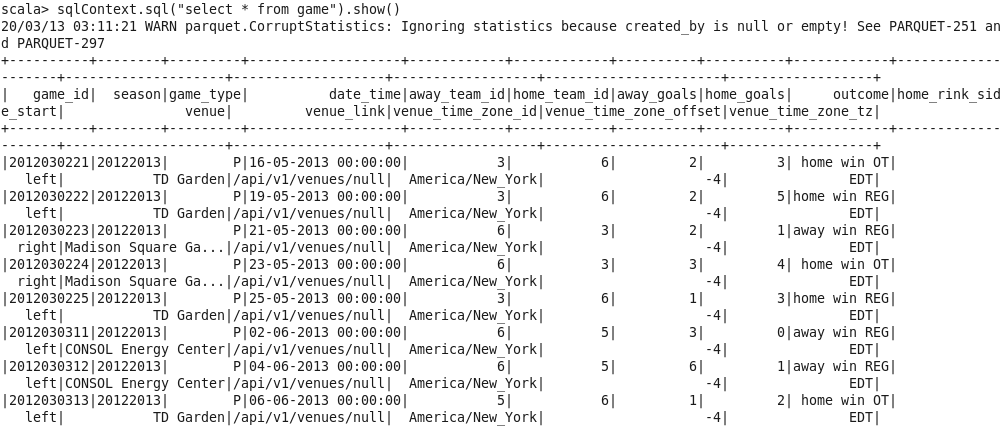
iv) val gameDataReady = mapGameData.toDF()

v) gameDataReady.write.parquet("/user/cloudera/game\_data3.parquet")

vi) val gameDataReady = sqlContext.read.parquet("/user/cloudera/game\_data3.parquet")

vii)gameDataReady.registerTempTable("game")

viii)sqlContext.sql("select \* from game").show()



2) clean\_game\_golie\_stats.csv

i) val gameGoalie = sc.textFile("/user/cloudera/clean\_game\_golie\_stats.csv")

ii) case class gamegolieData(game\_id: Long,player\_id: Long,team\_id: Integer,timeonice: Long,assists: Integer,goals: Integer,pim: Integer,shots: Integer,saves: Integer,powerplaysaves: Integer,shorthandedsaves: Integer,evensaves: Integer,shorthandedshotsagainst: Integer,evenshotsagainst: Integer,powerplayshotsagainst: Integer,decision: String,savepercentage:Float,powerplaysavepercentage: Float,evenstrengthsavepercentage: Float)

iii) val mapGameGolieData = gameGoalie.map(\_.split(",")).map(g => gamegolieData(g(0).toLong, g(1).toLong, g(2).toInt, g(3).toLong, g(4).toInt,g(5).toInt,g(6).toInt,g(7).toInt,g(8).toInt,g(9).toInt,g(10).toInt,g(11).toInt,g(12).toInt,g(13).toInt,g(14).toInt,g(15),g(16).toFloat,g(17).toFloat,g(18).toFloat))

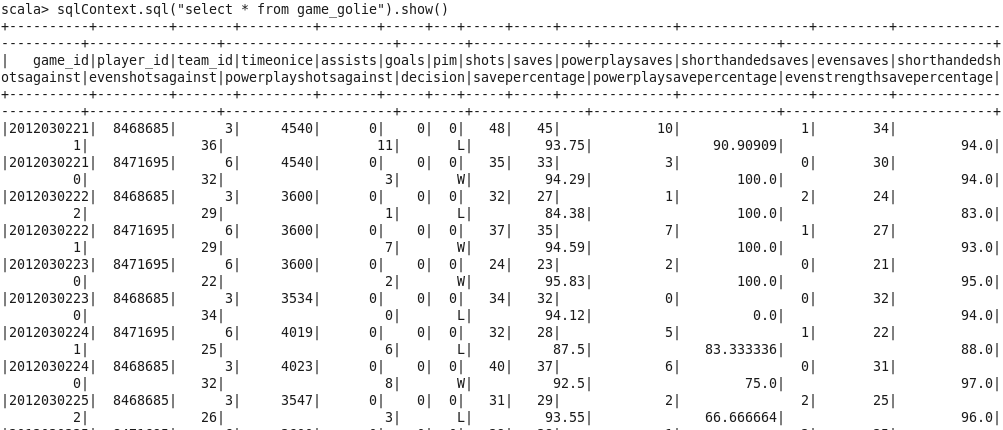
iv) val gameshiftsDataReady = mapGameGolieData.toDF()

v) gameshiftsdataReady.write.parquet("/user/cloudera/game\_data.parquet")

vi) val gameshiftsDataReady = sqlContext.read.parquet("/user/cloudera/game\_data.parquet")

vii)gameshiftsDataReady.registerTempTable("game\_golie")

viii)sqlContext.sql("select \* from game\_golie").show()



3) game\_skater.csv

i)val gameskaterFile = sc.textFile("/user/cloudera/game\_skater\_stats.csv")

ii)case class teamInfo(game\_id: Long,player\_id: Long,team\_id:Integer,timeonice: Long,assists:Integer,goals:Integer,shots:Integer,hits:Integer,powerplaygoals:Integer,powerplayassists: Integer,penaltyminutes: Integer,faceoffwins: Integer,faceofftaken: Integer,takeaways:Integer,giveaways: Integer,shorthandedgoals:Integer,shorthandedassists:Integer,bloacked: Integer,plusminus:Integer,eventimeonice:Integer,shorthandedtimeonice:Integer,powerplaytimeonice:Integer)

iii)val mapTeamInfo = gameskaterFile.map(\_.split(",")).map(t => teamInfo( t(0).toLong, t(1).toLong, t(2).toInt, t(3).toLong, t(4).toInt, t(5).toInt,t(6).toInt,t(7).toInt,t(8).toInt,t(9).toInt,t(10).toInt,t(11).toInt,t(12).toInt,t(13).toInt,t(14).toInt,t(15).toInt,t(16).toInt,t(17).toInt,t(18).toInt,t(19).toInt,t(20).toInt,t(21).toInt))

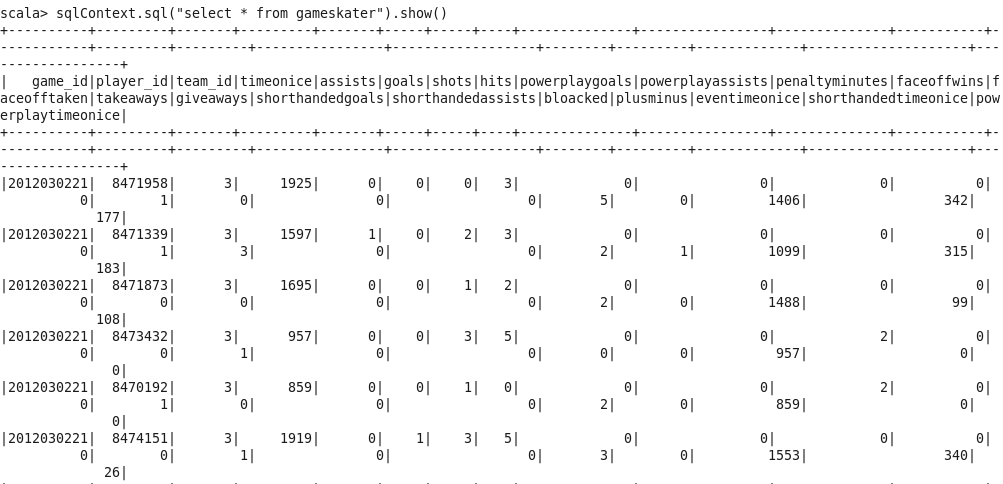
iv)val teamInfoReady = mapTeamInfo.toDF()

v)teamInfoReady.write.parquet("/user/cloudera/team\_info.parquet")

vi)val teamInfoReady = sqlContext.read.parquet("/user/cloudera/team\_info.parquet")

vii)teamInfoReady.registerTempTable("gameskater")

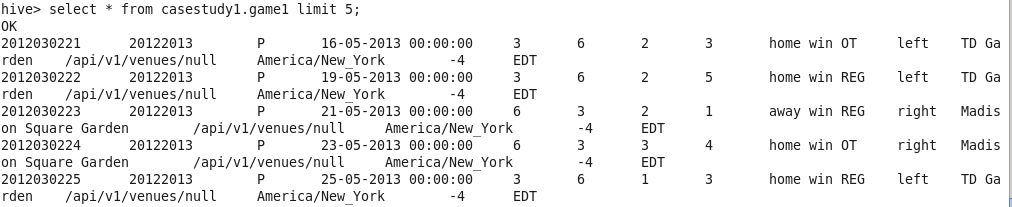
viii)sqlContext.sql("select \* from gameskater").show()



**Loading data into Hive table**

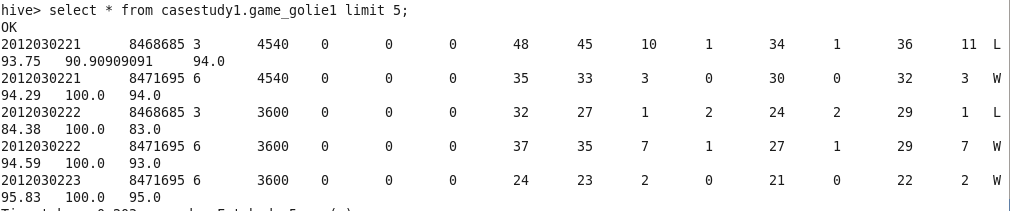
**Clean\_game.csv**

* hiveContext.sql("CREATE TABLE casestudy1.game1(game\_id bigint, season bigint, game\_type String, date\_time String,away\_team\_id int,home\_team\_id int,away\_goals int,home\_goals int,outcome String,home\_rink\_side\_start String,venue String,venue\_link String, venue\_time\_zone\_id String, venue\_time\_zone\_offset int,venue\_time\_zone\_tz String) row format delimited fields terminated by ','");
* hiveContext.sql("LOAD DATA INPATH '/user/cloudera/clean\_game.csv' INTO TABLE casestudy1.game1")



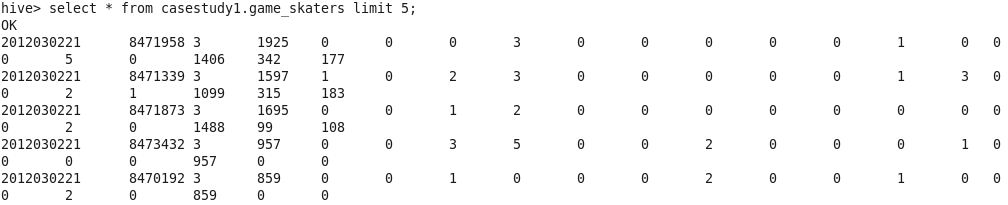
**clean\_game\_golie\_stats.csv**

* hiveContext.sql("CREATE TABLE casestudy1.game\_golie1(game\_id bigint,player\_id int,team\_id int,timeonice bigint,assists int,goals int,pim int,shot int,saves int,powerplaysaves int,shorthandedsaves int,evensaves int,shorthandedshotsagainst int,evenshotsagainst int,powerplayshotsagainst int,decision String,savepercentage double,powerplaysavepercentage double,evenstrengthsavepercentage double) row format delimited fields terminated by ','");
* hiveContext.sql("LOAD DATA INPATH '/user/cloudera/clean\_game\_golie\_stats.csv' INTO TABLE casestudy1.game\_golie1")



**game\_skater\_stats.csv**

* hiveContext.sql("CREATE TABLE casestudy1.game\_skaters(game\_id bigint,player\_id bigint,team\_id int,timeonice bigint,assists int,goals int,shots int,hits int,powerplaygoals int,powerplayassists int,penaltyminutes int,faceoffwins int,faceofftaken int,takeaways int,giveaways int,shorthandedgoals int,shorthandedassists int,bloacked int,plusminus int,eventimeonice int,shorthandedtimeonice int,powerplaytimeonice int) row format delimited fields terminated by ','")
* hiveContext.sql("LOAD DATA INPATH '/user/cloudera/game\_skater\_stats.csv' INTO TABLE casestudy1.game\_skaters")



**Join the game LEFT JOIN game\_goalie\_stats RIGHT JOIN game\_skaters**

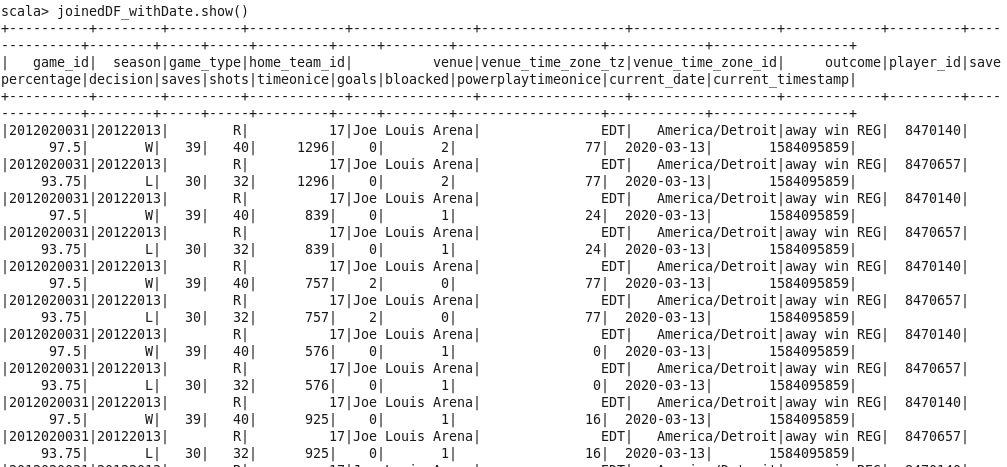
i)val resf = gameDataReady.join(gameshiftsDataReady ,Seq("game\_id"),"left")

ii) val leftTable = resf.select("season","game\_type","home\_team\_id","venue","venue\_time\_zone\_tz","venue\_time\_zone\_id","outcome","player\_id","savepercentage","decision","saves","shots","game\_id")

iii) val skater = teamInfoReady.select("game\_id","timeonice","goals","bloacked","powerplaytimeonice")

iv)val joinedDF = leftTable.join(skater,Seq("game\_id"),"right")

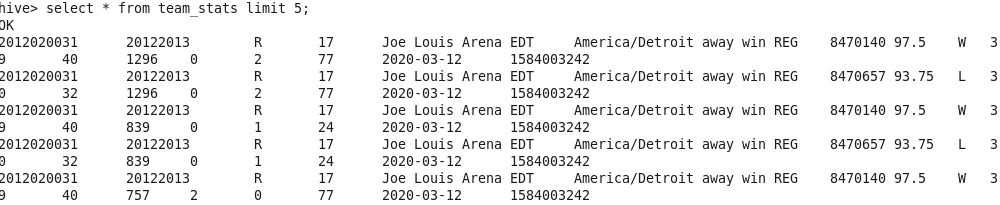
v) val joinedDF\_withDate = joinedDF.withColumn("current\_date",lit(current\_date())).withColumn("current\_timestamp",lit(u nix\_timestamp()))



**Loading Joined Table to HIVE :**

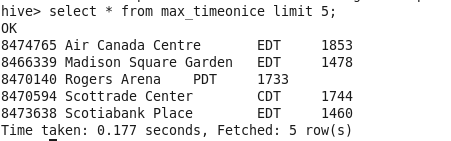
joinedDF\_withDate.registerTempTable("team\_stats")

sqlContext.sql("create table casestudy.team\_stats as select \* from team\_stats")

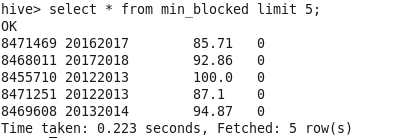


**6)** **Aggregations-5-6 aggregated tables (HiveQL)**

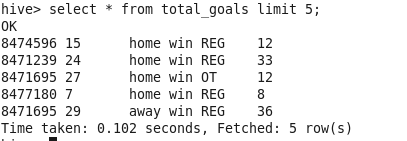
* hiveContext.sql("CREATE TABLE casestudy1.max\_timeonice as select player\_id,venue,venue\_time\_zone\_tz,max(timeonice) from casestudy.team\_stats group by player\_id,venue,venue\_time\_zone\_tz");



* hiveContext.sql("CREATE TABLE casestudy1.min\_blocked as select player\_id,season,savepercentage,min(bloacked) from casestudy.team\_stats group by player\_id,season,savepercentage");



* hiveContext.sql("CREATE TABLE casestudy1.total\_goals as select player\_id,shots,outcome,sum(goals) from casestudy.team\_stats group by player\_id,shots,outcome");



Select queries

---------------------------

1)Display the team\_stats joined target table data.

hiveContext.sql("select \* from casestudy.team\_stats limit 10").show();

game\_id| season|game\_type|home\_team\_id| venue|venue\_time\_zone\_tz|venue\_time\_zone\_id| outcome|player\_id|savepercentage|decision|saves|shots|timeonice|goals|bloacked|powerplaytimeonice|current\_date|current\_timestamp|

|2012020031|20122013| R| 17|Joe Louis Arena| EDT| America/Detroit|away win REG| 8470140| 97.5| W| 39| 40| 1296| 0| 2| 77| 2020-03-12| 1584003242|

|2012020031|20122013| R| 17|Joe Louis Arena| EDT| America/Detroit|away win REG| 8470657| 93.75| L| 30| 32| 1296| 0| 2| 77| 2020-03-12| 1584003242|

|2012020031|20122013| R| 17|Joe Louis Arena| EDT| America/Detroit|away win REG| 8470140| 97.5| W| 39| 40| 839| 0| 1| 24| 2020-03-12| 1584003242|

|2012020031|20122013| R| 17|Joe Louis Arena| EDT| America/Detroit|away win REG| 8470657| 93.75| L| 30| 32| 839| 0| 1| 24| 2020-03-12| 1584003242|

|2012020031|20122013| R| 17|Joe Louis Arena| EDT| America/Detroit|away win REG| 8470140| 97.5| W| 39| 40| 757| 2| 0| 77| 2020-03-12| 1584003242|

|2012020031|20122013| R| 17|Joe Louis Arena| EDT| America/Detroit|away win REG| 8470657| 93.75| L| 30| 32| 757| 2| 0| 77| 2020-03-12| 1584003242|

|2012020031|20122013| R| 17|Joe Louis Arena| EDT| America/Detroit|away win REG| 8470140| 97.5| W| 39| 40| 576| 0| 1| 0| 2020-03-12| 1584003242|

|2012020031|20122013| R| 17|Joe Louis Arena| EDT| America/Detroit|away win REG| 8470657| 93.75| L| 30| 32| 576| 0| 1| 0| 2020-03-12| 1584003242|

|2012020031|20122013| R| 17|Joe Louis Arena| EDT| America/Detroit|away win REG| 8470140| 97.5| W| 39| 40| 925| 0| 1| 16| 2020-03-12| 1584003242|

|2012020031|20122013| R| 17|Joe Louis Arena| EDT| America/Detroit|away win REG| 8470657| 93.75| L| 30| 32| 925| 0| 1| 16| 2020-03-12| 1584003242|

2)Display the game\_id which is having away as string included in it.

hiveContext.sql("select game\_id from casestudy.team\_stats where outcome like 'away%' limit 10").show();

| game\_id|

|2012020031|

|2012020031|

|2012020031|

|2012020031|

|2012020031|

|2012020031|

|2012020031|

|2012020031|

|2012020031|

|2012020031|

+----------+

3)Display game\_type from joined target table for all the game players in 20122013.

hiveContext.sql("select game\_type from casestudy.team\_stats where season = 20122013 group by game\_type ").show();

|game\_type|

| P|

| R|

4)Display maximum save percentage from min\_blocked table for each season

hiveContext.sql("select max(savepercentage) as savepercentage from casestudy1.min\_blocked group by season").show()

|savepercentage|

| 100.0|

| 100.0|

| 100.0|

| 100.0|

| 100.0|

| 100.0|

5)Display all the venues having timezone 'EDT'

hiveContext.sql("select venue from casestudy1.max\_timeonice where venue\_time\_zone\_tz like 'EDT'").show()

| venue|

| Air Canada Centre|

|Madison Square Ga...|

| Scotiabank Place|

| BB&T Center|

|CONSOL Energy Center|

| Capital One Arena|

|Tampa Bay Times F...|

| Verizon Center|

| Citi Field|

| Wells Fargo Center|

| PNC Arena|

| Wells Fargo Center|

| Barclays Center|

| Amalie Arena|

| Nassau Coliseum|

|Canadian Tire Centre|

|Little Caesars Arena|

| Nassau Coliseum|

|Tampa Bay Times F...|

| Nassau Coliseum|

6)Display player\_id in venue "PNC Arena"

hiveContext.sql("select player\_id from casestudy1.max\_timeonice where venue like 'PNC%'").show()

|player\_id|

| 8475717|

| 8473575|

| 8460704|

| 8473607|

| 8477465|

| 8470594|

| 8473523|

| 8475852|

| 8455710|

| 8475839|

| 8476839|

| 8471239|

| 8470645|

| 8471774|

| 8473503|

| 8460703|

| 8474690|

| 8471219|

| 8469548|

| 8460535|

7)Display player\_id who have shots greater than 20.

hiveContext.sql("select player\_id from casestudy1.total\_goals where shots>20").show()

|player\_id|

| 8471239|

| 8471695|

| 8471695|

| 8462147|

| 8470657|

| 8471695|

| 8477293|

| 8471679|

| 8471679|

| 8476839|

| 8471774|

| 8471695|

| 8470657|

| 8470147|

| 8470320|

| 8475311|

| 8475852|

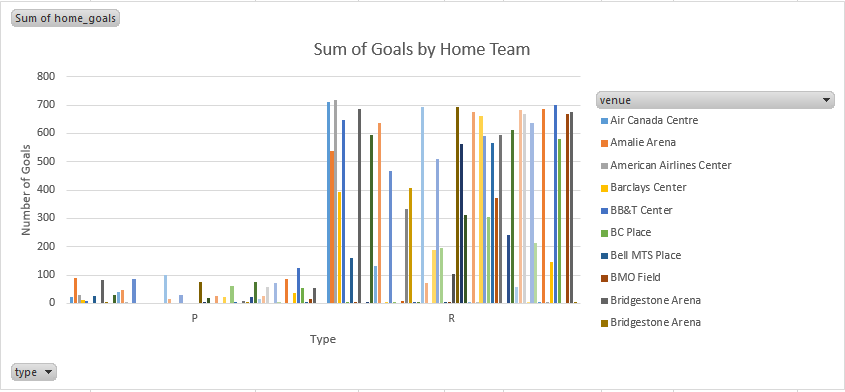
| 8475839|

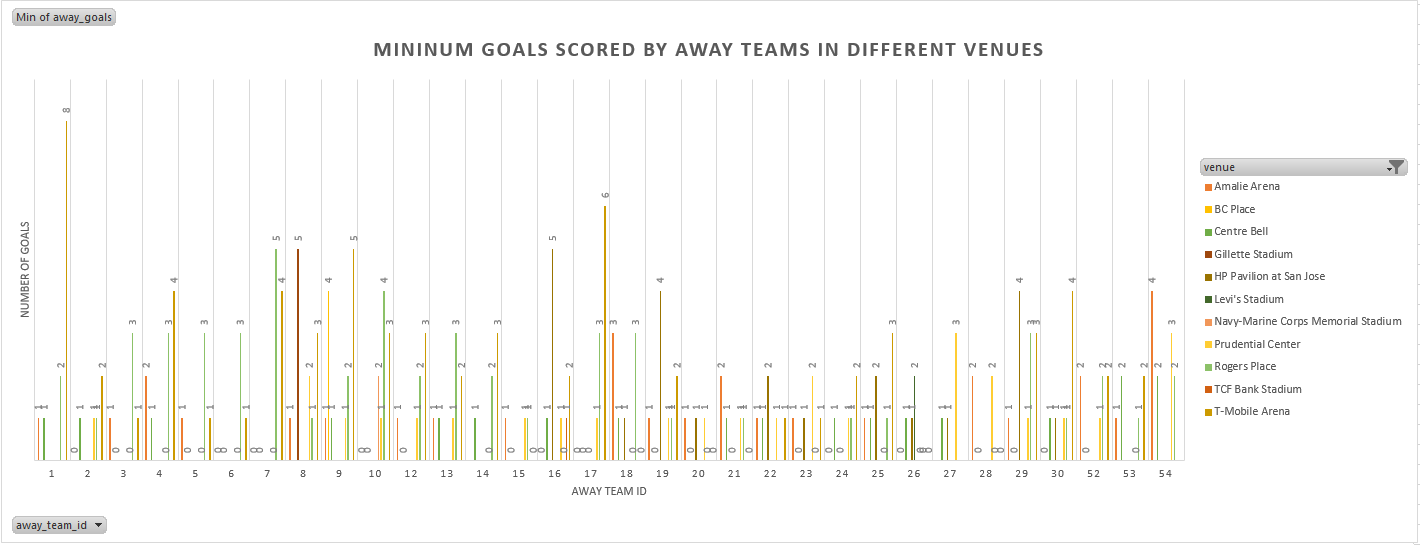
| 8470657|

| 8471306|

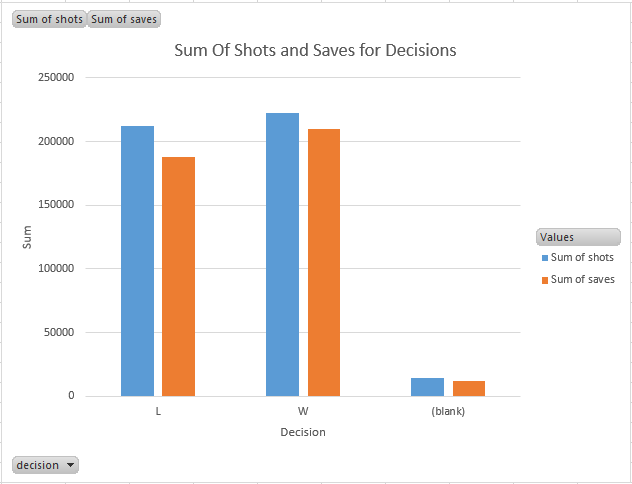
**7) BI-Visualization (Excel)**

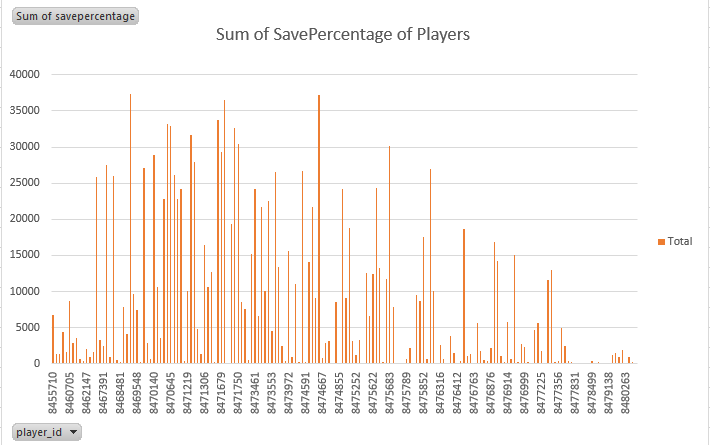
**clean\_game.csv**

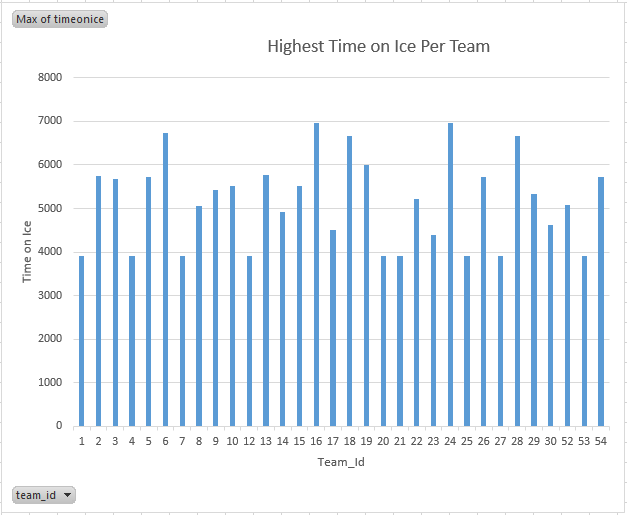
****

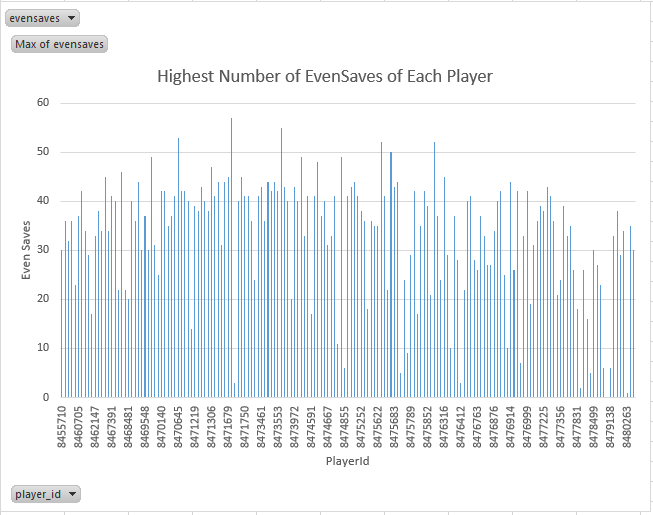
****

**clean\_game\_golie.csv**

****

****

****

****

**game\_skaters.csv**

