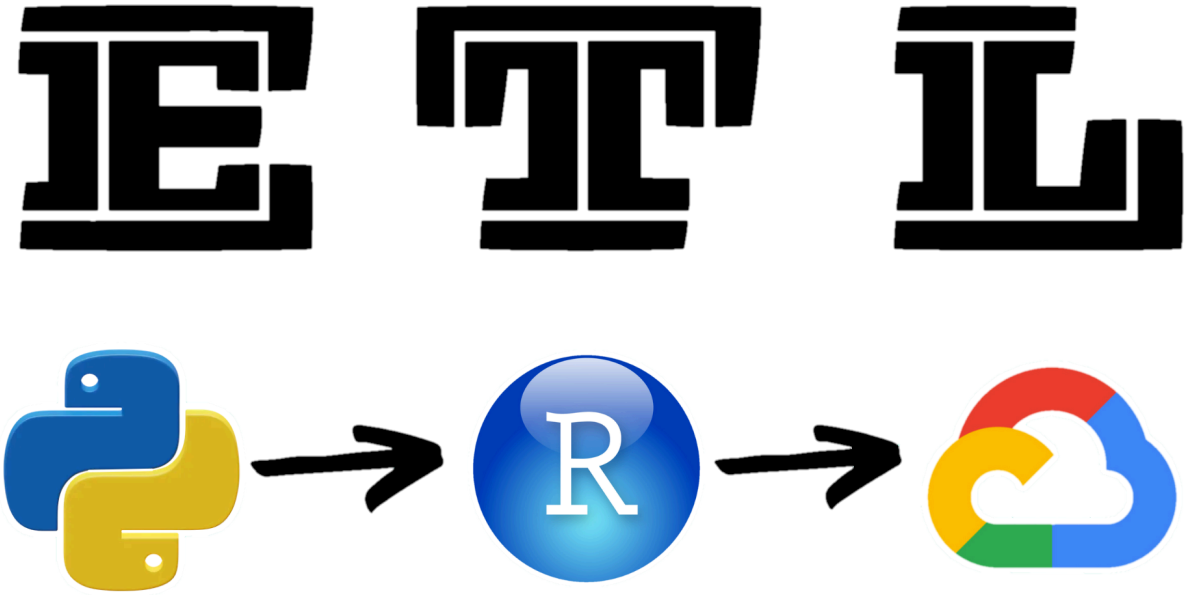


Data Warehousing (ETL) TRANSFORM

Sawandi Kirby

2024-05-15

Contents

1 ETL -(Extract Transform Load)

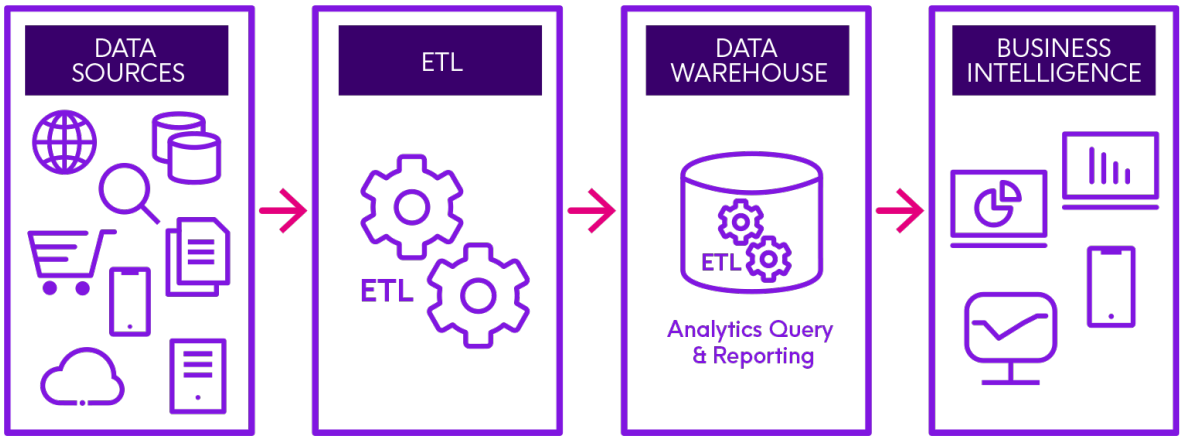


Figure 1: Extract Transform Load

1.1 Definition

1.1.1 ETL stands for "extract, transform, and load". It's a data integration process that combines data from multiple sources into a central repository, such as a data warehouse or data lake, and uses business rules to clean and organize it. The ETL process involves:

1.1.2 Extract: Raw data is copied or exported from source locations to a staging area. For example, data sources can include SQL or NoSQL servers, CRM and ERP systems, flat files, or email.

1.1.3 Transform: Cleansing, mapping, and transforming the raw data into a format that can be used by different applications.

1.1.4 Load: Writing the converted data from a staging area to a target database.

1.1.5 ETL enables data analysis to provide actionable business information, effectively preparing data for analysis and business intelligence processes. For example, an ETL tool might take updated accounting information from an ERP system (extract), combine it with other accounting data (transform), and store the transformed data in an organization's data lake for analytical analysis.

1.1.6 ETL pipelines are a set of tools and activities that move data from one system to another.

1.1.7 This is an overview of the data pipeline I've created for NBA Data scraped from the internet.

2 Extract

2.1 Python- Web Scrape

2.1.1 Web Scraping Code - The code to scrape NBA data can be found on GitHub on the Python Web Scrape Page

– Transform

3 Import data

3.1 Load Required Packages

3.1.1 Use function `install_packages` to install the required packages if you have not already.

3.1.2 Use function `library` to load the packages everytime R is restarted.

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ---------------------- tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(readxl)
library(openxlsx)
library(dplyr)
library(zoo)
```

```
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(fmsb)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
##
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
library(maps)
```

```
##
## Attaching package: 'maps'
##
## The following object is masked from 'package:purrr':
##
##     map
```

```
library(stringr)
library(ggplot2)
library(fuzzyjoin)
library(stringdist)
```

```
##
## Attaching package: 'stringdist'
##
## The following object is masked from 'package:tidyr':
##
##     extract
```

```r
library(stringr)
library(htmltools)
library(httpuv)
library(googleCloudStorageR)
```

```
## v Set default bucket name to 'kirby_studio_bucket'
```

```r
library(googleAuthR)
library(bigrquery)
```

3.2 Functions

3.3 Read excel file

3.4 We are going to build a few functions that will hopefully allow us to transform ths data a little easier.

3.4.1 Our Files are in an Excel format with each sheet representing a table.

3.4.2 Let's build a Function to read Excel file.

3.4.3 Get the names of all sheets in the Excel file.

3.4.4 Then read each sheet into a list of data frames.

```r
read_excel_file <- function(file_path) {
  sheet_names <- excel_sheets(file_path)

  all_data <- lapply(sheet_names, function(sheet) {
    read_excel(file_path, sheet = sheet)
  })

  return(all_data)
}
```

3.5 Trim Rows

3.5.1 We have data frames that contain 30 + columns, it would be very tedious to go throught each column with a trimrow.

3.5.2 So we are going to build a function to look for character columns, then to apply trimws to those columns in a data frame.

```r
trimws_df <- function(df) {
  char_cols <- sapply(df, is.character)
  df[char_cols] <- lapply(df[char_cols], trimws)
  return(df)
}
```

3.6 Convert to numeric

3.6.1 We know that there will be 0's and N/A values in our Latitude, Longitutde and Zipcode columns so let build a function to handle these special numeric columns.

3.6.2 First, define a placeholder value for missing or non-numeric coordinates.

3.6.3 Then, make a function to convert to numeric and handle non-numeric values

```
placeholder_value <- 0

convert_to_numeric <- function(column) {
  numeric_column <- as.numeric(as.character(column))
  numeric_column[is.na(numeric_column)] <- placeholder_value
  return(numeric_column)
}
```

3.7 Team Mapping

3.7.1 Define a named vector with team abbreviations and full names

```
team_mapping <- c(
  "atl" = "Atlanta Hawks",
  "bos" = "Boston Celtics",
  "cho" = "Charlotte Hornets",
  "chi" = "Chicago Bulls",
  "cle" = "Cleveland Cavaliers",
  "dal" = "Dallas Mavericks",
  "den" = "Denver Nuggets",
  "det" = "Detroit Pistons",
  "gsw" = "Golden State Warriors",
  "hou" = "Houston Rockets",
  "ind" = "Indiana Pacers",
  "lac" = "Los Angeles Clippers",
  "lal" = "Los Angeles Lakers",
  "mem" = "Memphis Grizzlies",
  "mia" = "Miami Heat",
  "mil" = "Milwaukee Bucks",
  "min" = "Minnesota Timberwolves",
  "nop" = "New Orleans Pelicans",
  "nyk" = "New York Knicks",
  "brk" = "Brooklyn Nets",
  "okc" = "Oklahoma City Thunder",
  "orl" = "Orlando Magic",
  "phi" = "Philadelphia 76ers",
  "pho" = "Phoenix Suns",
  "por" = "Portland Trail Blazers",
  "sas" = "San Antonio Spurs",
  "sac" = "Sacramento Kings",
  "tor" = "Toronto Raptors",
```

```
  "uta" = "Utah Jazz",
  "was" = "Washington Wizards"
)
```

3.8 Assign Location

3.8.1 There is a Home/Away column when we combine all of our data frames we can determine who was home and who was away.

3.8.2 we can build a function to assign Home/Away teams.

3.8.3 This function assigns Home and Away teams based on a boolean column.

```
assign_location <- function(data, boolean_column, Opponent.x, custom_value) {
  data <- data %>%
    mutate(Away = ifelse(!!rlang::ensym(boolean_column), Opponent.x, custom_value),
           Home = ifelse(!!rlang::ensym(boolean_column), custom_value, Opponent.x))
  return(data)
}
```

3.9 Height to Inches

3.9.1 Our Hieght column is in ft, in format. Easier for reading, harder for calculating.

3.9.2 This function will convert height columns to inches.

```
convert_height_to_inches <- function(df, column_name) {
  df <- df %>%
    mutate(Height = sapply(strsplit(df[[column_name]], "-"), function(x) as.integer(x[1]) * 12 + as.int
  return(df)
}
```

3.10 Match and replace College names

3.10.1 We have obtained a data frame containing details of all the colleges in the country including geo locations. Although our Roster data frames do not have a standard convention for naming the attended colleges of players.

3.10.2 We are going to have to build a function that can standerdize match and replace the college names so that we can merge our required columns.

3.10.3 First, we define common corrections.

3.10.4 Then , we standardize college names based on corrections list.

3.10.5 Then, we match and replace college names with a merge operation. # Applying corrections to roster_df. Standardize college names in coords_df. Lower case the values in bth data frames.

3.10.6 Next we are going to perform approximate matching. Using a threshold to avoid incorrect matches.

3.10.7 Finally, we update the College column with the matched values.

```
corrections <- list(
  "&"                          = "and",
  "alabama"                    = "the university of alabama",
  "arizona"                    = "university of arizona",
  "arizona state"              = "arizona state university campus immersion",
  "arkansas"                   = "university of arkansas",
  "auburn"                     = "auburn university",
  "ball state"                 = "ball state university",
  "baylor"                     = "baylor university",
  "belmont"                    = "belmont university",
  "boston"                     = "boston college",
  "bowling green"              = "bowling green state university-main campus",
  "buffalo"                    = "university at buffalo",
  "butler"                     = "butler university",
  "california"                 = "university of california, berkeley",
  "central florida"            = "university of central florida",
  "chattanooga"                = "the university of tennessee-chattanooga",
  "cincinnati"                 = "university of cincinnati-main campus",
  "clemson"                    = "clemson university",
  "colorado"                   = "university of colorado boulder",
  "colorado-colorado springs"  = "university of colorado colorado springs",
  "colorado state"             = "colorado state university-system office",
  "creighton"                  = "creighton university",
  "davidson"                   = "davidson college",
  "dayton"                     = "university of dayton",
  "denver"                     = "university of denver",
  "depaul"                     = "depaul university",
  "duke"                       = "duke university",
  "duquesne"                   = "duquesne university",
  "emu"                        = "eastern michigan university",
```

```
  "western kentucky"             = "western kentucky university",
  "wichita state"                = "wichita state university",
  "williams"                     = "williams college",
  "wisconsin"                    = "university of wisconsin-madison",
  "wku"                          = "western kentucky university",
  "wyoming"                      = "university of wyoming",
  "xavier"                       = "xavier university",
  "yakima valley"                = "yakima valley college"
)

standardize_college_name <- function(college_name, corrections) {
  if (tolower(college_name) %in% names(corrections)) {
    return(corrections[[tolower(college_name)]])
  } else {
    return(college_name)
  }
}

match_and_replace_colleges <- function(roster_df, coords_df, corrections, threshold = 0.05) {

  roster_df$College <- sapply(roster_df$College, standardize_college_name, corrections = corrections)

  coords_df <- coords_df %>%
    mutate(College = sapply(College, standardize_college_name, corrections = corrections),
           College_processed = tolower(gsub("\\s+", "", College)))

  roster_df <- roster_df %>%
    mutate(College_processed = tolower(gsub("\\s+", "", College)))

  matched_colleges <- sapply(roster_df$College_processed, function(college_name_processed) {
    if (!is.na(college_name_processed)) {
      distances <- stringdist::stringdist(college_name_processed, coords_df$College_processed, method =
      best_match_index <- which.min(distances)
      best_match_college <- coords_df$College[best_match_index]

      if (distances[best_match_index] < threshold) {
        return(best_match_college)
      } else {
        return(NA)
      }
    } else {
      return(NA)
    }
  })

  roster_df$College <- ifelse(is.na(matched_colleges), roster_df$College, matched_colleges)
  roster_df <- roster_df %>% select(-College_processed)

  return(roster_df)
}
```

3.11 Join Coordinates

3.11.1 Next, we have to Merge the columns frome our College df to our Roster data. Let use Left Join because we want all the data from from our Roster which will be on the left and only some data from the right which will be the colleges based on their match. The College name column.

3.11.2 Standardize college names in both data frames. Then, left join with specific columns from College_coords.

```
join_coordinates <- function(roster_df, coords_df) {
  roster_df$College <- sapply(roster_df$College, standardize_college_name, corrections)
  coords_df$College <- sapply(coords_df$College, standardize_college_name, corrections)

  merged_df <- left_join(roster_df, select(coords_df, College, Address, City, State, Zip, County, Latitu

  return(merged_df)
}
```

3.12 Assign Positions

3.12.1 Our last function will assign positions to players with missing values once we combine our data frames towards the end.

```
assign_positions <- function(per_game_df) {
  positions <- character(nrow(per_game_df))
  for (i in seq_along(per_game_df$Player)) {
    player <- per_game_df$Player[i]
    if (player %in% NBA_Roster$Player) {
      position <- NBA_Roster$Position[NBA_Roster$Player == player]
      positions[i] <- position
    } else {
      positions[i] <- NA
    }
  }
  per_game_df$Position <- positions
  return(per_game_df)
}
```

3.12.2 I'm going to repeat the function for te Totals df. I do not believe this is actually necessary, but better safe than sorry.

```
assign_positions_totals <- function(totals_df) {
  positions <- character(nrow(totals_df))
  for (i in seq_along(totals_df$Player)) {
    player <- totals_df$Player[i]
    if (player %in% NBA_Roster$Player) {
      position <- NBA_Roster$Position[NBA_Roster$Player == player]
```

```
      positions[i] <- position
    } else {
      positions[i] <- NA
    }
  }
  totals_df$Position <- positions
  return(totals_df)
}
```

3.13 Import Data

3.13.1 First, lets Import the college data and call it `College_coords`. Also, we have a df with Geolocation data for the countries around the world.

3.13.2 Let's import that as well.

```
College_coords <- read.csv("C:\\Users\\kirby\\OneDrive\\Documents\\Geo_Files\\Colleges_and_Universities

Country_coords <- read.csv("C:\\Users\\kirby\\OneDrive\\Documents\\Geo_Files\\country-coord.csv")
```

4 Additional Data

4.1 Country Coordinates

4.1.1 The data in the `Country_coords` df is formatted the same as our scraped basketball data. So we will not have to build functions for them.

4.1.2 Although we will change the names of some of the columns and trim rows to make them easier to work with.

```
Country_coords <- Country_coords %>%
  rename(Country.abr = Alpha.2.code, Country.abr3 = Alpha.3.code, Country.num = Numeric.code, Cty.Latitu

Country_coords$Country.abr<- trimws(tolower(Country_coords$Country.abr))
Country_coords$Country.abr3 <- trimws(Country_coords$Country.abr3)
```

4.2 College Coordinates

4.2.1 Let's do the same for our College data frame.

```
College_coords <- College_coords %>%
  rename(College = NAME, Address = ADDRESS, City = CITY, State = STATE, Zip = ZIP, County = COUNTY, Lati

College_coords$College <- trimws(tolower(College_coords$College))
College_coords$College <- trimws(College_coords$College)
```

4.2.2 As mentioned before, The `College_coords` df is a little more difficult do deal with. Let's go ahead and fix the data to what we need.

4.2.3 Convert columns to appropriate types. Handle missing values. Remove duplicate rows. Finally, rename columns if necessary (example: renaming "x" and "y" to "Longitude" and "Latitude")

4.2.4 Display the cleaned structure

```
College_coords <- College_coords %>%
  mutate(
    Zip = as.integer(Zip),
    TYPE = as.factor(TYPE),
    STATUS = as.factor(STATUS),
    County = as.factor(County),
    COUNTYFIPS = as.factor(COUNTYFIPS),
    COUNTRY = as.factor(COUNTRY),
    NAICS_CODE = as.factor(NAICS_CODE),
    SOURCEDATE = mdy_hms(SOURCEDATE, tz = "UTC"),
    VAL_DATE = mdy_hms(VAL_DATE, tz = "UTC"),
    STFIPS = as.factor(STFIPS),
    COFIPS = as.factor(COFIPS),
    SECTOR = as.factor(SECTOR),
    LEVEL = as.factor(LEVEL),
    CLOSE_DATE = as.factor(CLOSE_DATE),
    MERGE_ID = as.integer(MERGE_ID),
    ALIAS = as.factor(ALIAS),
    SIZE_SET = as.integer(SIZE_SET),
    INST_SIZE = as.integer(INST_SIZE),
    PT_ENROLL = as.integer(PT_ENROLL),
    FT_ENROLL = as.integer(FT_ENROLL),
    TOT_ENROLL = as.integer(TOT_ENROLL),
    HOUSING = as.integer(HOUSING),
    DORM_CAP = as.integer(DORM_CAP),
    TOT_EMP = as.integer(TOT_EMP),
    SHELTER_ID = as.factor(SHELTER_ID)
  )

College_coords <- College_coords %>%
  mutate(
    Zip = ifelse(Zip == -999, NA, Zip),
    ZIP4 = ifelse(ZIP4 == "NOT AVAILABLE", NA, ZIP4),
    TELEPHONE = ifelse(TELEPHONE == "NOT AVAILABLE", NA, TELEPHONE),
    ALIAS = ifelse(ALIAS == "NOT AVAILABLE", NA, ALIAS),
    DORM_CAP = ifelse(DORM_CAP == -999, NA, DORM_CAP),
    PT_ENROLL = ifelse(PT_ENROLL == -999, NA, PT_ENROLL),
    FT_ENROLL = ifelse(FT_ENROLL == -999, NA, FT_ENROLL),
    SIZE_SET = ifelse(SIZE_SET == -2, NA, SIZE_SET),
    CLOSE_DATE = ifelse(CLOSE_DATE == "-2", NA, CLOSE_DATE),
    SHELTER_ID = ifelse(SHELTER_ID == "NOT AVAILABLE", NA, SHELTER_ID)
  )

College_coords <- College_coords %>% distinct()
```

```r
College_coords <- College_coords %>%
  rename(
    Longitude.x = x,
    Latitude.y = y
  )

str(College_coords)
```

```
## 'data.frame':    6559 obs. of  45 variables:
##  $ OBJECTID   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ IPEDSID    : int  190752 481429 241739 455071 455141 455211 482185 243601 371052 131113 ...
##  $ College    : chr  "yeshiva of far rockaway derech ayson rabbinical seminary" "universal training i
##  $ Address    : chr  "802 HICKSVILLE RD" "174 JEFFERSON STREET" "1399 AVE. ANA G. MENDEZ" "10073 MAN
##  $ City       : chr  "FAR ROCKAWAY" "PERTH AMBOY" "SAN JUAN" "ST. LOUIS" ...
##  $ State      : chr  "NY" "NJ" "PR" "MO" ...
##  $ Zip        : int  11691 8861 926 63122 27514 89119 98109 778 36608 19720 ...
##  $ ZIP4       : chr  "5219" "4106" "2602" NA ...
##  $ TELEPHONE  : chr  "(718) 327-7600" "(732) 826-0155 EXT 106" "(787) 766-1717 EXT 6400" "(314) 647-8
##  $ TYPE       : Factor w/ 4 levels "-3","1","2","3": 3 4 3 4 4 4 3 3 4 3 ...
##  $ STATUS     : Factor w/ 7 levels "A","C","D","G",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ POPULATION : int  58 191 9913 146 241 298 153 14764 456 16867 ...
##  $ County     : Factor w/ 1044 levels "ABBEVILLE","ACADIA",..: 753 607 807 874 681 198 487 381 617 64
##  $ COUNTYFIPS : Factor w/ 1467 levels "01003","01015",..: 811 745 1458 680 874 723 1352 1447 22 161
##  $ COUNTRY    : Factor w/ 9 levels "ASM","FSM","GUM",..: 8 8 7 8 8 8 8 7 8 8 ...
##  $ Latitude   : num  40.6 40.5 18.4 38.6 35.9 ...
##  $ Longitude  : num  -73.7 -74.3 -66.1 -90.4 -79 ...
##  $ NAICS_CODE : Factor w/ 9 levels "611210","611310",..: 2 7 2 8 5 5 7 2 1 2 ...
##  $ NAICS_DESC : chr  "COLLEGES, UNIVERSITIES, AND PROFESSIONAL SCHOOLS" "OTHER TECHNICAL AND TRADE SC
##  $ SOURCE     : chr  "https://nces.ed.gov/collegenavigator/?id=190752" "https://nces.ed.gov/collegena
##  $ SOURCEDATE : POSIXct, format: "2015-08-10" "2015-08-10" ...
##  $ VAL_METHOD : chr  "IMAGERY" "IMAGERY/OTHER" "IMAGERY" "IMAGERY/OTHER" ...
##  $ VAL_DATE   : POSIXct, format: "2015-10-14" "2015-11-12" ...
##  $ WEBSITE    : chr  "https://www.yofr.org/" "www.universaluti.edu/" "cupey.uagm.edu/" "https://www.
##  $ STFIPS     : Factor w/ 57 levels "01","02","04",..: 33 31 55 26 34 29 48 55 1 8 ...
##  $ COFIPS     : Factor w/ 203 levels "001","003","005",..: 49 16 77 109 81 2 23 39 59 2 ...
##  $ SECTOR     : Factor w/ 11 levels "0","1","2","3",..: 3 10 3 10 10 10 6 3 10 3 ...
##  $ LEVEL      : Factor w/ 4 levels "-3","1","2","3": 2 2 3 4 3 3 3 3 4 2 ...
##  $ HI_OFFER   : int  30 0 12 0 0 0 0 11 0 12 ...
##  $ DEG_GRANT  : int  1 2 1 2 2 2 2 1 2 1 ...
##  $ LOCALE     : int  11 21 11 21 13 12 11 21 12 21 ...
##  $ CLOSE_DATE : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ MERGE_ID   : int  -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 ...
##  $ ALIAS      : int  NA NA 1749 NA NA 131 40 1750 638 NA ...
##  $ SIZE_SET   : int  24 NA 18 NA NA NA NA 17 10 17 ...
##  $ INST_SIZE  : int  1 1 3 1 1 1 1 4 1 4 ...
##  $ PT_ENROLL  : int  NA NA 2780 131 NA NA NA 4035 NA 10616 ...
##  $ FT_ENROLL  : int  43 176 6113 NA 216 268 135 9518 399 4153 ...
##  $ TOT_ENROLL : int  43 176 8893 131 216 268 135 13553 399 14769 ...
##  $ HOUSING    : int  1 2 2 2 2 2 2 2 2 2 ...
##  $ DORM_CAP   : int  35 NA NA NA NA NA NA NA NA NA ...
##  $ TOT_EMP    : int  15 15 1020 15 25 30 18 1211 57 2098 ...
##  $ SHELTER_ID : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ Longitude.x: num  -8209134 -8267688 -7353945 -10061915 -8797220 ...
```

```
##  $ Latitude.y : num  4953591 4940383 2082890 4664997 4290711 ...
```

4.2.5 Let's do a quick check to make sure our df is where we need it.

4.2.6 Step 1: Check geo location for valid range.

4.2.7 Step 2: Remove or handle missing values.

4.2.8 Step 3: Check for NaNs or Infinities.

4.2.9 Display the cleaned structure.

```
College_coords <- College_coords %>%
  filter(Longitude >= -180 & Longitude <= 180) %>%
  filter(Latitude >= -90 & Latitude <= 90)

College_coords <- College_coords %>%
  filter(!is.na(Longitude) & !is.na(Latitude))

College_coords <- College_coords %>%
  filter(!is.infinite(Longitude) & !is.infinite(Latitude))

str(College_coords)
```

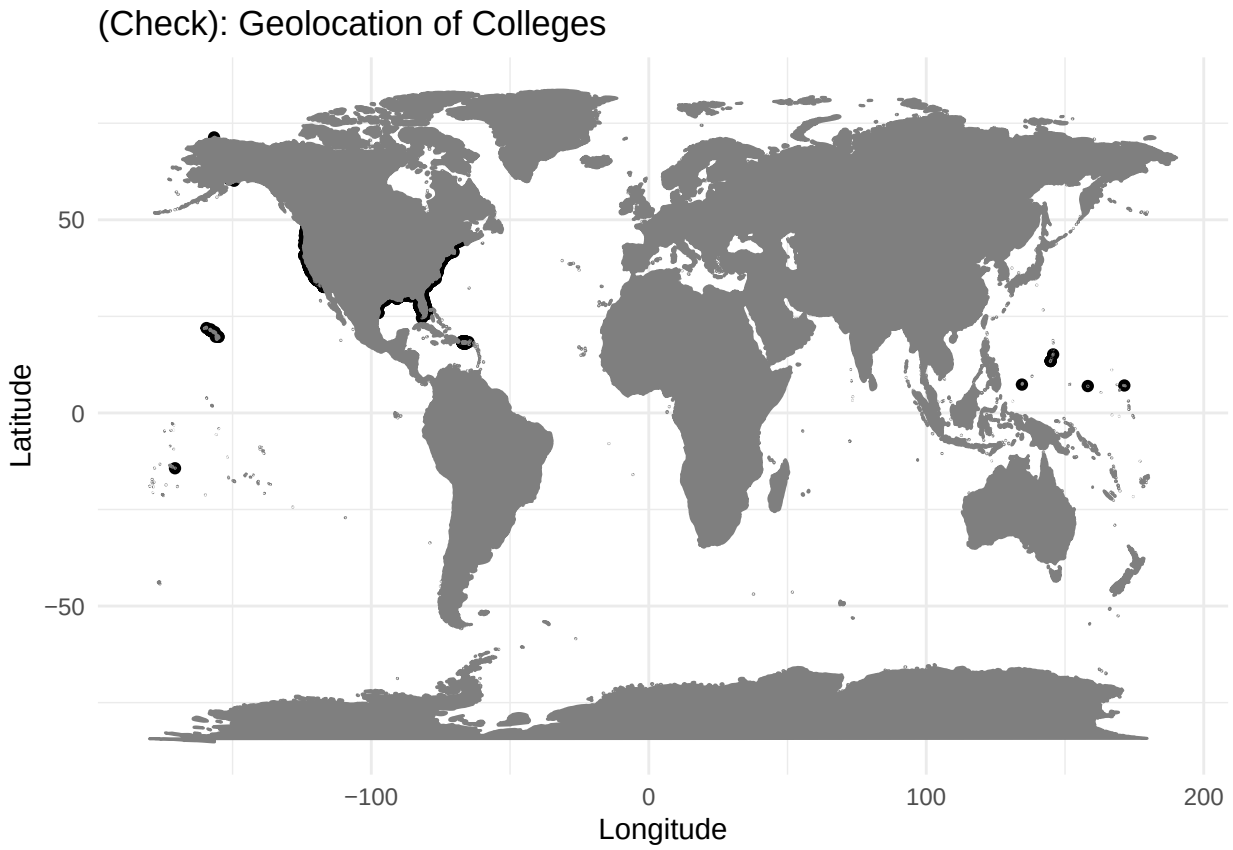
```
## 'data.frame':    6559 obs. of  45 variables:
##  $ OBJECTID   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ IPEDSID    : int  190752 481429 241739 455071 455141 455211 482185 243601 371052 131113 ...
##  $ College    : chr  "yeshiva of far rockaway derech ayson rabbinical seminary" "universal training
##  $ Address    : chr  "802 HICKSVILLE RD" "174 JEFFERSON STREET" "1399 AVE. ANA G. MENDEZ" "10073 MAN
##  $ City       : chr  "FAR ROCKAWAY" "PERTH AMBOY" "SAN JUAN" "ST. LOUIS" ...
##  $ State      : chr  "NY" "NJ" "PR" "MO" ...
##  $ Zip        : int  11691 8861 926 63122 27514 89119 98109 778 36608 19720 ...
##  $ ZIP4       : chr  "5219" "4106" "2602" NA ...
##  $ TELEPHONE  : chr  "(718) 327-7600" "(732) 826-0155 EXT 106" "(787) 766-1717 EXT 6400" "(314) 647-
##  $ TYPE       : Factor w/ 4 levels "-3","1","2","3": 3 4 3 4 4 4 3 3 4 3 ...
##  $ STATUS     : Factor w/ 7 levels "A","C","D","G",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ POPULATION : int  58 191 9913 146 241 298 153 14764 456 16867 ...
##  $ County     : Factor w/ 1044 levels "ABBEVILLE","ACADIA",..: 753 607 807 874 681 198 487 381 617 6
##  $ COUNTYFIPS : Factor w/ 1467 levels "01003","01015",..: 811 745 1458 680 874 723 1352 1447 22 161
##  $ COUNTRY    : Factor w/ 9 levels "ASM","FSM","GUM",..: 8 8 7 8 8 8 8 7 8 8 ...
##  $ Latitude   : num  40.6 40.5 18.4 38.6 35.9 ...
##  $ Longitude  : num  -73.7 -74.3 -66.1 -90.4 -79 ...
##  $ NAICS_CODE : Factor w/ 9 levels "611210","611310",..: 2 7 2 8 5 5 7 2 1 2 ...
##  $ NAICS_DESC : chr  "COLLEGES, UNIVERSITIES, AND PROFESSIONAL SCHOOLS" "OTHER TECHNICAL AND TRADE SC
##  $ SOURCE     : chr  "https://nces.ed.gov/collegenavigator/?id=190752" "https://nces.ed.gov/collegena
##  $ SOURCEDATE : POSIXct, format: "2015-08-10" "2015-08-10" ...
##  $ VAL_METHOD : chr  "IMAGERY" "IMAGERY/OTHER" "IMAGERY" "IMAGERY/OTHER" ...
##  $ VAL_DATE   : POSIXct, format: "2015-10-14" "2015-11-12" ...
##  $ WEBSITE    : chr  "https://www.yofr.org/" "www.universaluti.edu/" "cupey.uagm.edu/" "https://www.
##  $ STFIPS     : Factor w/ 57 levels "01","02","04",..: 33 31 55 26 34 29 48 55 1 8 ...
##  $ COFIPS     : Factor w/ 203 levels "001","003","005",..: 49 16 77 109 81 2 23 39 59 2 ...
```

```
##  $ SECTOR     : Factor w/ 11 levels "0","1","2","3",..: 3 10 3 10 10 10 6 3 10 3 ...
##  $ LEVEL      : Factor w/ 4 levels "-3","1","2","3": 2 2 3 4 3 3 3 3 4 2 ...
##  $ HI_OFFER   : int  30 0 12 0 0 0 0 11 0 12 ...
##  $ DEG_GRANT  : int  1 2 1 2 2 2 2 1 2 1 ...
##  $ LOCALE     : int  11 21 11 21 13 12 11 21 12 21 ...
##  $ CLOSE_DATE : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ MERGE_ID   : int  -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 ...
##  $ ALIAS      : int  NA NA 1749 NA NA 131 40 1750 638 NA ...
##  $ SIZE_SET   : int  24 NA 18 NA NA NA NA 17 10 17 ...
##  $ INST_SIZE  : int  1 1 3 1 1 1 1 4 1 4 ...
##  $ PT_ENROLL  : int  NA NA 2780 131 NA NA NA 4035 NA 10616 ...
##  $ FT_ENROLL  : int  43 176 6113 NA 216 268 135 9518 399 4153 ...
##  $ TOT_ENROLL : int  43 176 8893 131 216 268 135 13553 399 14769 ...
##  $ HOUSING    : int  1 2 2 2 2 2 2 2 2 2 ...
##  $ DORM_CAP   : int  35 NA NA NA NA NA NA NA NA NA ...
##  $ TOT_EMP    : int  15 15 1020 15 25 30 18 1211 57 2098 ...
##  $ SHELTER_ID : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ Longitude.x: num  -8209134 -8267688 -7353945 -10061915 -8797220 ...
##  $ Latitude.y : num  4953591 4940383 2082890 4664997 4290711 ...
```

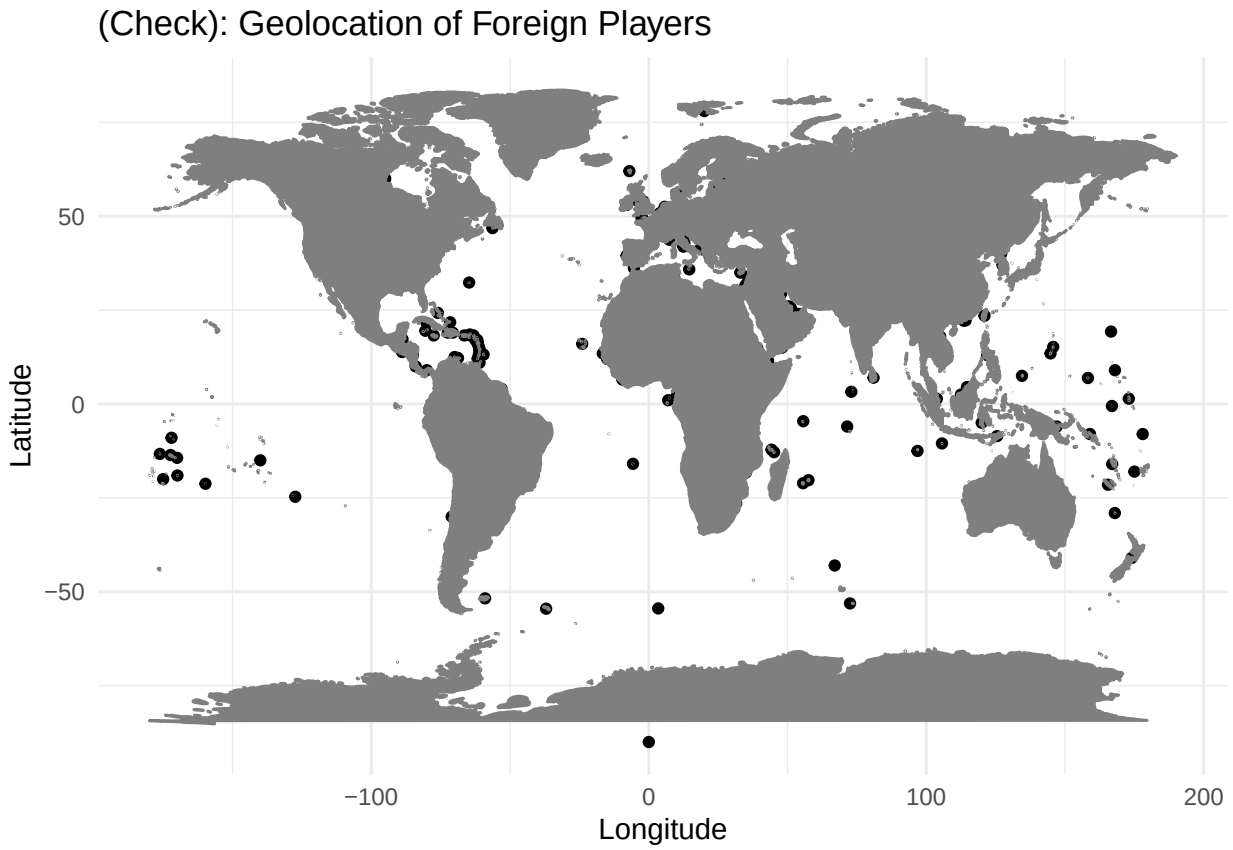
4.3 Verify

4.3.1 Optional Step 4: Visual inspection using a simple plot.

```
ggplot(College_coords, aes(x = Longitude, y = Latitude)) +
  geom_point() +
  borders("world", colour = "gray50", fill = "gray50") +
  theme_minimal() +
  labs(title = "(Check): Geolocation of Colleges", x = "Longitude", y = "Latitude")
```




```
ggplot(Country_coords, aes(x = Cty.Longitude, y = Cty.Latitude)) +
  geom_point() +
  borders("world", colour = "gray50", fill = "gray50") +
  theme_minimal() +
  labs(title = "(Check): Geolocation of Foreign Players", x = "Longitude", y = "Latitude")
```



4.4 Data

4.4.1 Finally, let's import or Scraped NBA Basketball Data.

```
ATL <- read_excel_file("C:\\Users\\kirby\\OneDrive\\Desktop\\NBA\\NBA_2024\\ATL_data.xlsx")
```

4.5 Name the sheet numbers from the list we created into dataframes

```
ATL_Roster <- ATL[[1]]
ATL_Per_Game <- ATL[[2]]
ATL_Totals <- ATL[[3]]
ATL_Starting_Lineup <- ATL[[4]]
ATL_Splits <- ATL[[5]]
ATL_Game_Log <- ATL[[6]]
ATL_Results <- ATL[[7]]
```

5 Clean Data

5.1 Working with the First Team (ATL)

5.1.1 First thig we are going to do is, replace empty strings with NA.

5.1.2 This will make them easier to find in our final check.

```
ATL_Roster <- ATL_Roster %>%
  mutate_if(is.character, ~if_else(. == "", NA, .))
ATL_Per_Game <- ATL_Per_Game %>%
  mutate_if(is.character, ~if_else(. == "", NA, .))
ATL_Totals <- ATL_Totals %>%
  mutate_if(is.character, ~if_else(. == "", NA, .))
ATL_Starting_Lineup <- ATL_Starting_Lineup %>%
  mutate_if(is.character, ~if_else(. == "", NA, .))
ATL_Splits <- ATL_Splits %>%
  mutate_if(is.character, ~if_else(. == "", NA, .))
ATL_Game_Log <- ATL_Game_Log %>%
  mutate_if(is.character, ~if_else(. == "", NA, .))
```

5.2 Roster

5.2.1 First start with our roster df

5.2.2 Lets get a glimpse of the dataframe and check out its structre

5.2.3 This will help us determine if our data columns are the correct data type

```
str(ATL_Roster)
```

```
## tibble [18 x 7] (S3: tbl_df/tbl/data.frame)
##  $ Player      : chr [1:18] "Bogdan Bogdanović" "Dejounte Murray" "Clint Capela" "Garrison Mathews"
##  $ Position    : chr [1:18] "sg" "sg" "c" "sg" ...
##  $ Height      : chr [1:18] "6-5" "6-5" "6-10" "6-5" ...
##  $ Weight      : num [1:18] 220 180 240 215 215 225 220 235 164 240 ...
##  $ BirthDate   : POSIXct[1:18], format: "1992-08-18" "1996-09-19" ...
##  $ BirthCountry: chr [1:18] "rs" "us" "ch" "us" ...
##  $ College     : chr [1:18] NA "washington" NA "lipscomb" ...
```

```
head(ATL_Roster)
```

```
## # A tibble: 6 x 7
##   Player      Position Height Weight BirthDate           BirthCountry College
##   <chr>       <chr>    <chr>   <dbl> <dttm>              <chr>        <chr>
## 1 Bogdan Bog~ sg       6-5       220 1992-08-18 00:00:00 rs           <NA>
## 2 Dejounte M~ sg       6-5       180 1996-09-19 00:00:00 us           washin~
## 3 Clint Cape~ c        6-10      240 1994-05-18 00:00:00 ch           <NA>
```

```
## 4 Garrison M~ sg       6-5       215 1996-10-24 00:00:00 us           lipsco~
## 5 Saddiq Bey  sf       6-7       215 1999-04-09 00:00:00 us           villan~
## 6 De'Andre H~ sf       6-8       225 1997-12-02 00:00:00 us           virgin~
```

5.2.4 Even though we have a good list of Player names lets get the first and last into their own columns and make the string all lowercase so the system can read them better.

5.2.5 First remove (TW) tag from Player names

```
ATL_Roster$Player <- gsub("\\(TW\\)", "", ATL_Roster$Player)

ATL_Roster <- ATL_Roster %>%
  separate(Player, into = c("FirstName", "LastName"), sep = " ", remove = FALSE)
```

5.2.5.1 Then use the separate function

```
## Warning: Expected 2 pieces. Additional pieces discarded in 3 rows [13, 16,
## 17].
```

5.2.6 Now we can we can use `trimws` function to first remove the whitespace around our string,.

5.2.7 Then use the `tolower` function to transform our string values into all lowecase letters.

```
ATL_Roster$Player <- trimws(tolower(ATL_Roster$Player))
ATL_Roster$FirstName <- trimws(tolower(ATL_Roster$FirstName))
ATL_Roster$LastName <- trimws(tolower(ATL_Roster$LastName))
```

5.2.8 We see that the columns that should be 'chr' are characters, 'num' are numeric, and 'POSIXct' is for date/time.

5.2.9 The Height column is in character format although it would be easier to calculate if they were in numeric format.

5.2.10 So lets use our function to convert the height column.

```
ATL_Roster <- convert_height_to_inches(ATL_Roster, 'Height')
```

5.2.11 Now it's time to merge our Roster df with our College geo locations.

5.2.12 First we are going to insert "none" as our place holder for college names of Players that did not attend a US college.

```
ATL_Roster$College[is.na(ATL_Roster$College)] <- "none"
head(ATL_Roster$College, n = 20)
```

```
##  [1] "none"             "washington"       "none"
##  [4] "lipscomb"         "villanova"        "virginia"
##  [7] "duke"             "usc"              "oklahoma"
## [10] "maryland"         "florida state"    "marquette"
## [13] "none"             "duke"             "michigan"
## [16] "penn state"       "belmont"          "washington state"
```

5.2.13 Next, apply the function to match and replace college names with standardization.

```
ATL_Roster <- match_and_replace_colleges(ATL_Roster, College_coords, corrections)
head(ATL_Roster$College, n = 20)
```

```
##                                                 none
##                                               "none"
##                                           washington
##                                         "washington"
##                                                 none
##                                               "none"
##                             lipscomb.lipscomb university
##                                    "lipscomb university"
##                           villanova.villanova university
##                                  "villanova university"
##             virginia.university of virginia-main campus
##                    "university of virginia-main campus"
##                                     duke.duke university
##                                        "duke university"
##                    usc.university of southern california
##                      "university of southern california"
##           oklahoma.university of oklahoma-norman campus
##                  "university of oklahoma-norman campus"
##            maryland.university of maryland-college park
##                   "university of maryland-college park"
##                  florida state.florida state university
##                               "florida state university"
##                           marquette.marquette university
##                                  "marquette university"
##                                                 none
##                                               "none"
##                                     duke.duke university
##                                        "duke university"
##               michigan.university of michigan-ann arbor
##                      "university of michigan-ann arbor"
## penn state.pennsylvania state university-main campus
##             "pennsylvania state university-main campus"
##                               belmont.belmont university
##                                    "belmont university"
##                                        washington state
##                                      "washington state"
```

5.2.14 Then, we are going to use our `join_coordinates` to left_join our data.

```
ATL_Roster <- join_coordinates(ATL_Roster, College_coords)
head(ATL_Roster)
```

```
## # A tibble: 6 x 16
##   Player        FirstName LastName Position Height Weight BirthDate
##   <chr>         <chr>     <chr>    <chr>     <dbl>  <dbl> <dttm>
## 1 bogdan bogda~ bogdan    bogdano~ sg           77    220 1992-08-18 00:00:00
## 2 dejounte mur~ dejounte  murray   sg           77    180 1996-09-19 00:00:00
## 3 clint capela  clint     capela   c            82    240 1994-05-18 00:00:00
## 4 garrison mat~ garrison  mathews  sg           77    215 1996-10-24 00:00:00
## 5 saddiq bey    saddiq    bey      sf           79    215 1999-04-09 00:00:00
## 6 de'andre hun~ de'andre  hunter   sf           80    225 1997-12-02 00:00:00
## # i 9 more variables: BirthCountry <chr>, College <chr>, Address <chr>,
## #   City <chr>, State <chr>, Zip <int>, County <fct>, Latitude <dbl>,
## #   Longitude <dbl>
```

5.2.15 Next, let's change the name of BirthCountry column to County.abr to match the `Country_coords` df so that we can use the merge function.

```
ATL_Roster <- ATL_Roster %>%
  rename(Country.abr = BirthCountry)

ATL_Roster <- merge(ATL_Roster, Country_coords, by = "Country.abr")
```

5.2.16 Now that we have brought over college data we have to find a way to handle the NA values that arise from player with n US college attendance.

```
ATL_Roster$Address[is.na(ATL_Roster$Address)] <- "none"
ATL_Roster$City[is.na(ATL_Roster$City)] <- "none"
ATL_Roster$State[is.na(ATL_Roster$State)] <- "na"
ATL_Roster <- ATL_Roster %>%
  mutate(County = as.character(County))
ATL_Roster$County[is.na(ATL_Roster$County)] <- "other"
```

5.2.17 Finally, apply the function to the Zip column and apply the function to Longitude and Latitude column.

```
ATL_Roster <- ATL_Roster %>%
  mutate(
    Zip = convert_to_numeric(Zip)
  )

ATL_Roster <- ATL_Roster %>%
```

```
  mutate(
    Longitude = convert_to_numeric(Longitude),
    Latitude = convert_to_numeric(Latitude)
  )
```

5.2.18 Review the df

5.2.19 Make sure our changes took affect

```
str(ATL_Roster)
```

```
## 'data.frame':    18 obs. of  21 variables:
##  $ Country.abr  : chr  "ao" "ch" "cz" "rs" ...
##  $ Player       : chr  "bruno fernando" "clint capela" "vit krejci" "bogdan bogdanović" ...
##  $ FirstName    : chr  "bruno" "clint" "vit" "bogdan" ...
##  $ LastName     : chr  "fernando" "capela" "krejci" "bogdanović" ...
##  $ Position     : chr  "c" "c" "pg" "sg" ...
##  $ Height       : num  81 82 80 77 83 77 81 77 79 80 ...
##  $ Weight       : num  240 240 195 220 210 180 220 215 215 225 ...
##  $ BirthDate    : POSIXct, format: "1998-08-15" "1994-05-18" ...
##  $ College      : chr  "university of maryland-college park" "none" "none" "none" ...
##  $ Address      : chr  "NOT AVAILABLE" "none" "none" "none" ...
##  $ City         : chr  "COLLEGE PARK" "none" "none" "none" ...
##  $ State        : chr  "MD" "na" "na" "na" ...
##  $ Zip          : num  20742 0 0 0 0 ...
##  $ County       : chr  "PRINCE GEORGE'S" "other" "other" "other" ...
##  $ Latitude     : num  39 0 0 0 0 ...
##  $ Longitude    : num  -76.9 0 0 0 0 ...
##  $ Country      : chr  "Angola" "Switzerland" "Czech Republic" "Serbia" ...
##  $ Country.abr3 : chr  "AGO" "CHE" "CZE" "SRB" ...
##  $ Country.num  : int  24 756 203 688 686 840 840 840 840 840 ...
##  $ Cty.Latitude : num  -12.5 47 49.8 44 14 ...
##  $ Cty.Longitude: num  18.5 8 15.5 21 -14 -97 -97 -97 -97 -97 ...
```

```
head(ATL_Roster)
```

```
##   Country.abr            Player FirstName   LastName Position Height Weight
## 1          ao    bruno fernando     bruno   fernando        c     81    240
## 2          ch      clint capela     clint     capela        c     82    240
## 3          cz        vit krejci       vit     krejci       pg     80    195
## 4          rs bogdan bogdanović    bogdan bogdanović       sg     77    220
## 5          sn    mouhamed gueye  mouhamed      gueye       pf     83    210
## 6          us   dejounte murray  dejounte     murray       sg     77    180
##    BirthDate                              College       Address         City
## 1 1998-08-15 university of maryland-college park NOT AVAILABLE COLLEGE PARK
## 2 1994-05-18                                 none          none         none
## 3 2000-06-19                                 none          none         none
## 4 1992-08-18                                 none          none         none
## 5 2002-11-09                   washington state          none         none
## 6 1996-09-19                         washington          none         none
```

```
##   State   Zip          County Latitude Longitude        Country Country.abr3
## 1    MD 20742 PRINCE GEORGE'S 38.98861  -76.9397         Angola          AGO
## 2    na     0           other  0.00000    0.0000    Switzerland          CHE
## 3    na     0           other  0.00000    0.0000 Czech Republic          CZE
## 4    na     0           other  0.00000    0.0000         Serbia          SRB
## 5    na     0           other  0.00000    0.0000        Senegal          SEN
## 6    na     0           other  0.00000    0.0000  United States          USA
##   Country.num Cty.Latitude Cty.Longitude
## 1          24       -12.50          18.5
## 2         756        47.00           8.0
## 3         203        49.75          15.5
## 4         688        44.00          21.0
## 5         686        14.00         -14.0
## 6         840        38.00         -97.0
```

5.3 Per Game

5.3.1 Now we can move on to the next df, Per_Game.

5.3.2 Start off by getting a view of the df and look at the structure.

```
str(ATL_Per_Game)
```

```
## tibble [19 x 27] (S3: tbl_df/tbl/data.frame)
##  $ Player                          : chr [1:19] "trae young" "dejounte murray" "jalen johnson" "sa
##  $ Age                             : num [1:19] 25 27 22 24 31 26 29 23 23 25 ...
##  $ GamesPlayedPerGame              : chr [1:19] "54" "78" "56" "63" ...
##  $ GamesStarted                    : num [1:19] 54 78 52 51 33 37 73 8 14 2 ...
##  $ MinutesPlayedPerGame            : num [1:19] 36 35.7 33.7 32.7 30.4 29.5 25.8 25.5 24.6 15.2 ..
##  $ FieldGoalsPerGame               : num [1:19] 8 8.6 6.4 4.6 6 5.3 4.8 4.1 2.3 2.4 ...
##  $ FieldGoalAttemptsPerGame        : num [1:19] 18.7 18.8 12.5 11.1 13.9 11.6 8.5 6.6 4.7 4.2 ...
##  $ FieldGoalPercentPerGame         : num [1:19] 0.43 0.459 0.511 0.416 0.428 0.459 0.571 0.611 0.4
##  $ ThreePointFieldGoalsPerGame     : num [1:19] 3.2 2.6 1.3 1.8 3 2.1 0 0.4 1.3 0 ...
##  $ ThreePointFieldGoalAttemptsPerGame: num [1:19] 8.7 7.1 3.6 5.7 8.1 5.3 0 1.3 3.1 0.1 ...
##  $ ThreePointFieldGoalPercentPerGame : num [1:19] 0.373 0.363 0.355 0.316 0.374 0.385 0 0.333 0.412 0
##  $ TwoPointFieldGoalsPerGame       : num [1:19] 4.8 6 5.1 2.8 2.9 3.3 4.8 3.6 1 2.4 ...
##  $ TwoPointFieldGoalAttemptsPerGame  : num [1:19] 10 11.7 9 5.4 5.8 6.3 8.5 5.4 1.6 4.1 ...
##  $ TwoPointFieldGoalPercentPerGame   : num [1:19] 0.479 0.518 0.574 0.522 0.503 0.521 0.572 0.676 0.6
##  $ EffectiveFieldGoalPercentPerGame  : num [1:19] 0.516 0.528 0.562 0.497 0.537 0.547 0.571 0.642 0.6
##  $ FreeThrowsPerGame               : num [1:19] 6.4 2.7 1.9 2.7 1.9 2.9 1.8 1.7 0.2 1.4 ...
##  $ FreeThrowAttemptsPerGame        : num [1:19] 7.5 3.4 2.6 3.2 2.1 3.4 2.8 2.1 0.3 2.1 ...
##  $ FreeThrowPercentPerGame         : num [1:19] 0.855 0.794 0.728 0.837 0.921 0.847 0.631 0.793 0.8
##  $ OffensiveReboundsPerGame        : num [1:19] 0.4 0.8 1.3 2.7 0.7 0.5 4.6 2.6 0.5 1.3 ...
##  $ DefensiveReboundsPerGame        : num [1:19] 2.3 4.5 7.4 3.9 2.8 3.4 6 4.2 1.9 3 ...
##  $ TotalReboundsPerGame            : num [1:19] 2.8 5.3 8.7 6.5 3.4 3.9 10.6 6.8 2.4 4.3 ...
##  $ AssistsPerGame                  : num [1:19] 10.8 6.4 3.6 1.5 3.1 1.5 1.2 1.3 2.3 1 ...
##  $ StealsPerGame                   : num [1:19] 1.3 1.4 1.2 0.8 1.2 0.7 0.6 0.5 0.6 0.6 ...
##  $ BlocksPerGame                   : num [1:19] 0.2 0.3 0.8 0.2 0.3 0.3 1.5 1.1 0.3 0.6 ...
##  $ TurnoversPerGame                : num [1:19] 4.4 2.6 1.8 0.9 1.4 1.5 1 0.8 0.9 1 ...
##  $ PersonalFoulsPerGame            : num [1:19] 2 1.8 2.4 1.4 2.3 2.6 2.2 2.9 2.2 2.4 ...
##  $ PointsPerGame                   : num [1:19] 25.7 22.5 16 13.7 16.9 15.6 11.5 10.2 6.1 6.3 ...
```

```
head(ATL_Per_Game)
```

```
## # A tibble: 6 x 27
##   Player             Age GamesPlayedPerGame GamesStarted MinutesPlayedPerGame
##   <chr>            <dbl> <chr>                      <dbl>                <dbl>
## 1 trae young          25 54                            54                 36
## 2 dejounte murray     27 78                            78                 35.7
## 3 jalen johnson       22 56                            52                 33.7
## 4 saddiq bey          24 63                            51                 32.7
## 5 bogdan bogdanov~    31 79                            33                 30.4
## 6 de'andre hunter     26 57                            37                 29.5
## # i 22 more variables: FieldGoalsPerGame <dbl>,
## #   FieldGoalAttemptsPerGame <dbl>, FieldGoalPercentPerGame <dbl>,
## #   ThreePointFieldGoalsPerGame <dbl>,
## #   ThreePointFieldGoalAttemptsPerGame <dbl>,
## #   ThreePointFieldGoalPercentPerGame <dbl>,
## #   TwoPointFieldGoalsPerGame <dbl>, TwoPointFieldGoalAttemptsPerGame <dbl>,
## #   TwoPointFieldGoalPercentPerGame <dbl>, ...
```

5.3.3 Its seems that the only incorrect column in this data frame is the GamesPlayed column.

5.3.4 So lets go ahead and fix that.

```
ATL_Per_Game$GamesPlayed <-as.numeric(ATL_Per_Game$GamesPlayedPerGame)
```

5.4 Totals

5.4.1 Let's remove blank rows from the `Totals` df.

```
ATL_Totals <- ATL_Totals[-c(20), ]
```

5.5 Game Log

5.5.1 Lets check our Game_Log

```
str(ATL_Game_Log)
```

```
## tibble [91 x 37] (S3: tbl_df/tbl/data.frame)
##  $ Date                : POSIXct[1:91], format: NA "2023-10-25" ...
##  $ Opp                 : chr [1:91] NA "cho" "nyk" "mil" ...
##  $ WinLoss             : chr [1:91] NA "L" "L" "W" ...
##  $ Points              : num [1:91] NA 110 120 127 127 130 123 117 120 109 ...
##  $ OppPoints           : num [1:91] NA 116 126 110 113 121 105 126 119 117 ...
##  $ FieldGoals          : num [1:91] NA 39 42 47 48 46 45 38 41 38 ...
##  $ FieldGoalAttempts   : num [1:91] NA 93 87 93 86 92 93 102 85 87 ...
```

5.5.2 As we can see all of the data types seems to be correct.

5.5.3 Although we can go ahead and make sure that the entire Date column is in the same format as the first couple of values we can see.

```
ATL_Game_Log <- ATL_Game_Log[complete.cases(ATL_Game_Log$Date), ]
ATL_Game_Log$Date <- as.Date(ATL_Game_Log$Date, format = "%Y-%m-%d")
```

5.5.4 Also lets turn the WinLoss column into a boolean because we only have two possible values that can stand for TRUE and FALSE

```
ATL_Game_Log$WinLoss <-ifelse(ATL_Game_Log$WinLoss == "W", TRUE, FALSE)
```

5.5.5 In our `Game_Log` df we have our Opponents abbreviated. we can link their results and better determine the Opponents performance by linking them between the `Game_Log` and `Rsults`.

5.5.6 First we will match the abbreviations with the teams full name using our team mapping.

```
# Merge team names into ATL_Game_Log
ATL_Game_Log <- ATL_Game_Log %>%
  mutate(Opponent = team_mapping[Opp])
```

5.6 Starting Lineups

5.6.1 Let's make sure our changes took effect

5.6.2 We can continue to use `head` and `str`

```
str(ATL_Starting_Lineup)
```

```
## tibble [82 x 13] (S3: tbl_df/tbl/data.frame)
##  $ Date          : chr [1:82] "2023-10-25" "2023-10-27" "2023-10-29" "2023-10-30" ...
##  $ Start(ET)     : logi [1:82] NA NA NA NA NA NA ...
##  $ _             : logi [1:82] NA NA NA NA NA NA ...
##  $ BoxScore      : chr [1:82] "Box Score" "Box Score" "Box Score" "Box Score" ...
##  $ HomeAway      : chr [1:82] "@" NA "@" NA ...
##  $ Opponent      : chr [1:82] "Charlotte Hornets" "New York Knicks" "Milwaukee Bucks" "Minnesota Timb
##  $ WinLoss       : chr [1:82] "L" "L" "W" "W" ...
##  $ Overtime      : chr [1:82] NA NA NA NA ...
##  $ TeamPoints    : chr [1:82] "110" "120" "127" "127" ...
##  $ OpponentPoints: chr [1:82] "116" "126" "110" "113" ...
##  $ Wins          : chr [1:82] "0" "0" "1" "2" ...
##  $ Losses        : chr [1:82] "1" "2" "2" "2" ...
##  $ StartingLineup: chr [1:82] "S. Bey · C. Capela · D. Hunter · D. Murray · T. Young" "S. Bey · C. Ca
```

```
head(ATL_Starting_Lineup)
```

```
## # A tibble: 6 x 13
##   Date       `Start(ET)` `_`   BoxScore  HomeAway Opponent   WinLoss Overtime
##   <chr>      <lgl>       <lgl> <chr>     <chr>    <chr>      <chr>   <chr>
## 1 2023-10-25 NA          NA    Box Score @        Charlotte~ L       <NA>
## 2 2023-10-27 NA          NA    Box Score <NA>     New York ~ L       <NA>
## 3 2023-10-29 NA          NA    Box Score @        Milwaukee~ W       <NA>
## 4 2023-10-30 NA          NA    Box Score <NA>     Minnesota~ W       <NA>
## 5 2023-11-01 NA          NA    Box Score <NA>     Washingto~ W       <NA>
## 6 2023-11-04 NA          NA    Box Score @        New Orlea~ W       <NA>
## # i 5 more variables: TeamPoints <chr>, OpponentPoints <chr>, Wins <chr>,
## #   Losses <chr>, StartingLineup <chr>
```

5.7 This df seems to have a few issues that need to be attended to.

5.7.1 First we need to fill in the NA values in the HomeAway and Overtime columns.

5.7.2 Then, because they are one of two valus we can convert them into boolean values, including the WinLoss column.

```
ATL_Starting_Lineup$HomeAway <- ifelse(is.na(ATL_Starting_Lineup$HomeAway), "Home", ATL_Starting_Lineup$
ATL_Starting_Lineup$Overtime <- ifelse(is.na(ATL_Starting_Lineup$Overtime), "rt", ATL_Starting_Lineup$O

ATL_Starting_Lineup$HomeAway <- ifelse(ATL_Starting_Lineup$HomeAway == "Home", TRUE, FALSE)
ATL_Starting_Lineup$Overtime <- ifelse(ATL_Starting_Lineup$Overtime == "ot", TRUE, FALSE)
ATL_Starting_Lineup$WinLoss <- ifelse(ATL_Starting_Lineup$WinLoss == "W", TRUE, FALSE)
```

5.7.3 Then we see there are multiple columns entirely with NA values, 'Start(ET)', '_', and 'BoxScore'.

5.7.4 We can use the `subset` function in order to accomplish this.

```
ATL_Starting_Lineup <- subset(ATL_Starting_Lineup, select = -`Start(ET)`)
ATL_Starting_Lineup <- subset(ATL_Starting_Lineup, select = -`_`)
ATL_Starting_Lineup <- subset(ATL_Starting_Lineup, select = -`BoxScore`)
```

5.7.5 Now we can convert our columns with number values into numeric.

5.7.6 And make sure our Date column is formatted correctly.

```
ATL_Starting_Lineup$TeamPoints <- as.numeric(ATL_Starting_Lineup$TeamPoints)
ATL_Starting_Lineup$OpponentPoints <- as.numeric(ATL_Starting_Lineup$OpponentPoints)
ATL_Starting_Lineup$Wins <- as.numeric(ATL_Starting_Lineup$Wins)
ATL_Starting_Lineup$Losses <- as.numeric(ATL_Starting_Lineup$Losses)

ATL_Starting_Lineup$Date <- as.Date(ATL_Starting_Lineup$Date, format = "%Y-%m-%d")
```

5.7.7 We need to find a way to link the `Starting_Lineups` df to the `Roster`, `Per_Game` and `Totals` dfs.

5.7.8 Because we have the starting players we can link them through the players last name. Let's go ahead and seperate the ame into FirstName and LastName.

```
ATL_Starting_Lineup <- ATL_Starting_Lineup %>%
  separate_rows(StartingLineup, sep = " · ")

ATL_Starting_Lineup <- ATL_Starting_Lineup %>%
    separate(StartingLineup, into = c("FirstName", "LastName"), sep = " ", remove = FALSE)

ATL_Starting_Lineup$FirstName <- trimws(tolower(ATL_Starting_Lineup$FirstName))
ATL_Starting_Lineup$LastName <- trimws(tolower(ATL_Starting_Lineup$LastName))
```

5.8 Results

5.8.1 Now it's time to check out the Results df

```
str(ATL_Results)
```

```
## tibble [87 x 7] (S3: tbl_df/tbl/data.frame)
##  $ Date     : POSIXct[1:87], format: "2023-10-25" "2023-10-27" ...
##  $ HomeAway : chr [1:87] "@" NA "@" NA ...
##  $ Opponent : chr [1:87] "charlotte hornets" "new york knicks" "milwaukee bucks" "minnesota timberwo
##  $ Winloss  : chr [1:87] "L" "L" "W" "W" ...
##  $ Overtime : chr [1:87] NA NA NA NA ...
##  $ Points   : num [1:87] 110 120 127 127 130 123 117 120 109 126 ...
##  $ OppPoints: num [1:87] 116 126 110 113 121 105 126 119 117 120 ...
```

```
head(ATL_Results)
```

```
## # A tibble: 6 x 7
##   Date                HomeAway Opponent     Winloss Overtime Points OppPoints
##   <dttm>              <chr>    <chr>        <chr>   <chr>     <dbl>     <dbl>
## 1 2023-10-25 00:00:00 @        charlotte h~ L       <NA>        110       116
## 2 2023-10-27 00:00:00 <NA>     new york kn~ L       <NA>        120       126
## 3 2023-10-29 00:00:00 @        milwaukee b~ W       <NA>        127       110
## 4 2023-10-30 00:00:00 <NA>     minnesota t~ W       <NA>        127       113
## 5 2023-11-01 00:00:00 <NA>     washington ~ W       <NA>        130       121
## 6 2023-11-04 00:00:00 @        new orleans~ W       <NA>        123       105
```

5.8.2 This df has more possible boolean columns that could be converted and a Date column

5.8.3 If you look closely you can see that there are entire rows of NA values.

```
ATL_Results$Date <- as.Date(ATL_Results$Date, format = "%Y-%m-%d")
ATL_Results$HomeAway <- ifelse(is.na(ATL_Results$HomeAway), "Home", ATL_Results$HomeAway)
ATL_Results$Overtime <- ifelse(is.na(ATL_Results$Overtime), "rt", ATL_Results$Overtime)

ATL_Results$HomeAway <- ifelse(ATL_Results$HomeAway == "Home", TRUE, FALSE)
ATL_Results$Overtime <- ifelse(ATL_Results$Overtime == "ot", TRUE, FALSE)
ATL_Results$Winloss <- ifelse(ATL_Results$Winloss == "W", TRUE, FALSE)
ATL_Results <- ATL_Results[-c(21,42,63,84), ]
```

5.8.3.1 We can go ahead and take those away

5.8.4 Now we can merge our Opponent Teams performance.

5.8.5 Step 1: Calculate Opponent Win-Loss Records.Ensure to handle missing values. Ensure no leading/trailing whitespace in column names.

5.8.6 Step 2: Merge win-loss records with ATL_Results. Lowercase ATL_Game_Log Opponent column.

5.8.7 Step 3: Merge ATL_Game_Log with enhanced ATL_Results.

5.8.8 Step 4: Categorize Opponents into Tiers.

5.8.9 Finally, display the resulting data frame.

```
Opponent_Records <- ATL_Results %>%
  group_by(Opponent) %>%
  summarize(
    OpponentWins = sum(!Winloss, na.rm = TRUE),
    OpponentLosses = sum(Winloss, na.rm = TRUE)
  )

names(ATL_Results) <- trimws(names(ATL_Results))
names(Opponent_Records) <- trimws(names(Opponent_Records))
names(ATL_Game_Log) <- trimws(names(ATL_Game_Log))

ATL_Results <- ATL_Results %>%
  left_join(Opponent_Records, by = "Opponent")

ATL_Game_Log$Opponent <- tolower(ATL_Game_Log$Opponent)

ATL_Game_Log <- ATL_Game_Log %>%
  left_join(ATL_Results %>% select(Date, Opponent, OpponentWins, OpponentLosses), by = c("Date", "Oppon

ATL_Game_Log <- ATL_Game_Log %>%
  mutate(OpponentStrength = case_when(
    OpponentWins / (OpponentWins + OpponentLosses) >= 0.6 ~ "Strong",
    OpponentWins / (OpponentWins + OpponentLosses) <= 0.4 ~ "Weak",
    TRUE ~ "Average"
```

```
  ))

print(ATL_Game_Log)
```

```
## # A tibble: 82 x 41
##    Date       Opp   WinLoss Points OppPoints FieldGoals FieldGoalAttempts
##    <date>     <chr> <lgl>    <dbl>     <dbl>      <dbl>             <dbl>
##  1 2023-10-25 cho   FALSE      110       116         39                93
##  2 2023-10-27 nyk   FALSE      120       126         42                87
##  3 2023-10-29 mil   TRUE       127       110         47                93
##  4 2023-10-30 min   TRUE       127       113         48                86
##  5 2023-11-01 was   TRUE       130       121         46                92
##  6 2023-11-04 nop   TRUE       123       105         45                93
##  7 2023-11-06 okc   FALSE      117       126         38               102
##  8 2023-11-09 orl   TRUE       120       119         41                85
##  9 2023-11-11 mia   FALSE      109       117         38                87
## 10 2023-11-14 det   TRUE       126       120         45                87
## # i 72 more rows
## # i 34 more variables: FieldGoalPercent <dbl>, ThreePoints <dbl>,
## #   ThreePointAttempts <dbl>, ThreePointPercent <dbl>, FreeThrows <dbl>,
## #   FreeThrowAttempts <dbl>, FreeThrowPercent <dbl>,
## #   OffensiveRebounds <dbl>, TotalRebounds <dbl>, Assists <dbl>,
## #   Steals <dbl>, Blocks <dbl>, Turnovers <dbl>, PersonalFouls <dbl>,
## #   OppFieldGoals <dbl>, OppFieldGoalAttempts <dbl>, ...
```

```
print(ATL_Results)
```

```
## # A tibble: 83 x 9
##    Date       HomeAway Opponent             Winloss Overtime Points OppPoints
##    <date>     <lgl>    <chr>                <lgl>   <lgl>     <dbl>     <dbl>
##  1 2023-10-25 FALSE    charlotte hornets    FALSE   FALSE       110       116
##  2 2023-10-27 TRUE     new york knicks      FALSE   FALSE       120       126
##  3 2023-10-29 FALSE    milwaukee bucks      TRUE    FALSE       127       110
##  4 2023-10-30 TRUE     minnesota timberwol~ TRUE    FALSE       127       113
##  5 2023-11-01 TRUE     washington wizards   TRUE    FALSE       130       121
##  6 2023-11-04 FALSE    new orleans pelicans TRUE    FALSE       123       105
##  7 2023-11-06 FALSE    oklahoma city thund~ FALSE   FALSE       117       126
##  8 2023-11-09 FALSE    orlando magic        TRUE    FALSE       120       119
##  9 2023-11-11 TRUE     miami heat           FALSE   FALSE       109       117
## 10 2023-11-14 FALSE    detroit pistons      TRUE    FALSE       126       120
## # i 73 more rows
## # i 2 more variables: OpponentWins <int>, OpponentLosses <int>
```

5.9 Trim Rows

5.9.1 Since we have built the function to trim all rows of our data frames, we're going to run all of our data frames through it.

```
ATL_Roster <- trimws_df(ATL_Roster)
ATL_Game_Log <- trimws_df(ATL_Game_Log)
```

```
ATL_Per_Game <- trimws_df(ATL_Per_Game)
ATL_Starting_Lineup <- trimws_df(ATL_Starting_Lineup)
ATL_Splits <- trimws_df(ATL_Splits)
ATL_Results <- trimws_df(ATL_Results)
ATL_Totals <- trimws_df(ATL_Totals)
```

5.10 Team Tiers

5.10.1 Let's calculate the each teams win rate percentage.

5.10.2 Then, categorize the team's strength

```
team_win_rate <- ATL_Results %>%
  summarize(TeamWins = sum(Winloss, na.rm = TRUE),
            TeamLosses = sum(!Winloss, na.rm = TRUE)) %>%
  mutate(WinRate = TeamWins / (TeamWins + TeamLosses))

ATL_team_tier <- team_win_rate %>%
  mutate(TeamStrength = case_when(
    WinRate >= 0.6 ~ "Strong",
    WinRate <= 0.4 ~ "Weak",
    TRUE ~ "Average"
  ))

print(ATL_team_tier)
```

```
## # A tibble: 1 x 4
##   TeamWins TeamLosses WinRate TeamStrength
##      <int>      <int>   <dbl> <chr>
## 1       36         47   0.434 Average
```

5.10.3 Repeat these steps for all of our team Excel files, BOS, BRK, CHI, etc...

5.11 Bind Rows

5.11.1 Now we use `bind_rows` to combine and stack the data frames into one big df.

```
NBA_Roster <- bind_rows(ATL_Roster, BOS_Roster, BRK_Roster, CHA_Roster, CHI_Roster, CLE_Roster, DAL_Ros
```

5.11.2 Let's give the teams an index to help identify each team in the df by their abbreviation "ATL", "BOS", etc...

5.11.3 First define a data frame containing team names and their respective data frames. Then bind the rows of all team results. And add a column indicating the team.

5.11.4 Finally we're going to add a Date column to help identify the year this df is going to represent. Then Check the df.

5.11.5 Define a data frame containing team names and their respective data frames.

5.11.6 Bind the rows of all team results.

5.11.7 Add a column indicating the team.

```
team_data <- tibble(
  Team = c("ATL", "BOS", "BRK", "CHA", "CHI", "CLE", "DAL", "DEN", "DET", "GSW", "HOU", "IND",
           "LAC", "LAL", "MEM", "MIA", "MIL", "MIN", "NOP", "NYK", "OKC", "ORL",
           "PHI", "PHO", "POR", "SAC", "SAS", "TOR", "UTA", "WAS"),
  Roster = list(ATL_Roster, BOS_Roster, BRK_Roster, CHA_Roster, CHI_Roster, CLE_Roster, DAL_Roster,
                DEN_Roster, DET_Roster, GSW_Roster, HOU_Roster, IND_Roster,
                LAC_Roster, LAL_Roster, MEM_Roster, MIA_Roster, MIL_Roster,
                MIN_Roster, NOP_Roster, NYK_Roster, OKC_Roster, ORL_Roster,
                PHI_Roster, PHO_Roster, POR_Roster, SAC_Roster, SAS_Roster,
                TOR_Roster, UTA_Roster, WAS_Roster)
)

NBA_Roster <- bind_rows(team_data$Roster, .id = "Team_Index")

NBA_Roster <- NBA_Roster %>%
  mutate(Team = team_data$Team[as.numeric(Team_Index)])
```

5.11.8 continue this process...

5.11.9 Although, remember to call the function to assign positions to NBA_Per_Game and NBA_Totals.

```
NBA_Per_Game <- assign_positions(NBA_Per_Game)

NBA_Per_Game <- NBA_Per_Game %>%
  mutate(Team = team_data$Team[as.numeric(Team_Index)])
```

5.11.10 continue this process for the `Totals`, `Starting_Lineups`, `Game_Log`, `Results` and `Team_Tiers` dfs....

5.11.11 Also, remember to bind the rows of all team results.

5.11.12 For the `Game_Log` df let's remember to add a column to signify a team Win and a team Loss for custom data visualization.

```
NBA_Game_Log <- NBA_Game_Log %>%
  mutate(TeamWinLoss = ifelse(WinLoss, paste0(Team, "Win"), paste0(Team, "Loss")))
```

5.11.13 And for the `Results` df let's remember to add a column to signify a team Home and a team Away for custom data visualization.

```
NBA_Results <- NBA_Results %>%
  mutate(TeamColor = ifelse(HomeAway, paste0(Team, "Home"), paste0(Team, "Away")))
```

5.11.14 Define a data frame containing team names and their respective data frames.

```
NBA_team_tier <- bind_rows(ATL_team_tier, BOS_team_tier, BRK_team_tier, CHA_team_tier, CHI_team_tier, C
```

5.11.15 Bind the rows of all team results.

5.11.16 Add a column indicating the team.

```
team_data <- tibble(
  Team = c("ATL", "BOS", "BRK", "CHA", "CHI", "CLE", "DAL", "DEN", "DET", "GSW", "HOU", "IND",
           "LAC", "LAL", "MEM", "MIA", "MIL", "MIN", "NOP", "NYK", "OKC", "ORL",
           "PHI", "PHO", "POR", "SAC", "SAS", "TOR", "UTA", "WAS"),
  Team_Tier = list(ATL_team_tier, BOS_team_tier, BRK_team_tier, CHA_team_tier, CHI_team_tier, CLE_team_
                   DEN_team_tier, DET_team_tier, GSW_team_tier, HOU_team_tier, IND_team_tier,
                   LAC_team_tier, LAL_team_tier, MEM_team_tier, MIA_team_tier, MIL_team_tier,
                   MIN_team_tier, NOP_team_tier, NYK_team_tier, OKC_team_tier, ORL_team_tier,
                   PHI_team_tier, PHO_team_tier, POR_team_tier, SAC_team_tier, SAS_team_tier,
                   TOR_team_tier, UTA_team_tier, WAS_team_tier)
)

NBA_team_tier <- bind_rows(team_data$Team_Tier, .id = "Team_Index")

NBA_team_tier <- NBA_team_tier %>%
  mutate(Team = team_data$Team[as.numeric(Team_Index)])

NBA_team_tier$Team_Index <- as.numeric(NBA_team_tier$Team_Index)
```

5.12 Remove Objects

5.13 Use the `rm` function.

5.13.1 Now that we are no longer working with these data frames, we can go ahead and remove them using the `rm` function to accomplish this.

```
rm(ATL_Game_Log, ATL_Roster, ATL_Per_Game, ATL_Results, ATL_Starting_Lineup, ATL_Totals, ATL_Splits, ATL
```

```
## Warning in rm(CHO_Game_Log, CHO_Roster, CHO_Per_Game, CHO_Results,
## CHO_Starting_Lineup, : object 'CHO_Game_Log' not found

## Warning in rm(CHO_Game_Log, CHO_Roster, CHO_Per_Game, CHO_Results,
## CHO_Starting_Lineup, : object 'CHO_Roster' not found

## Warning in rm(CHO_Game_Log, CHO_Roster, CHO_Per_Game, CHO_Results,
## CHO_Starting_Lineup, : object 'CHO_Per_Game' not found

## Warning in rm(CHO_Game_Log, CHO_Roster, CHO_Per_Game, CHO_Results,
## CHO_Starting_Lineup, : object 'CHO_Results' not found

## Warning in rm(CHO_Game_Log, CHO_Roster, CHO_Per_Game, CHO_Results,
## CHO_Starting_Lineup, : object 'CHO_Starting_Lineup' not found

## Warning in rm(CHO_Game_Log, CHO_Roster, CHO_Per_Game, CHO_Results,
## CHO_Starting_Lineup, : object 'CHO_Totals' not found

## Warning in rm(CHO_Game_Log, CHO_Roster, CHO_Per_Game, CHO_Results,
## CHO_Starting_Lineup, : object 'CHO_Splits' not found

## Warning in rm(CHO_Game_Log, CHO_Roster, CHO_Per_Game, CHO_Results,
## CHO_Starting_Lineup, : object 'CHO' not found
```

6 Multi-Layered Check

6.1 Roster

6.1.1 Let's run our data frames through a detialed scan to make sure we all of our columns are clean and formatted.

6.1.2 Check class of each column.

6.1.3 Check summary statistics for each numeric column.

6.1.4 Check for NA values in each column.

6.1.5 Check for non-numeric values in each column.

6.1.6 Identify rows with NA values in each column.

6.1.7 Identify rows with non-numeric values in each numeric column.

6.1.8 Display rows with non-numeric values in each numeric column.

```
for (col in names(NBA_Roster)) {
  cat("Column:", col, ", Class:", class(NBA_Roster[[col]]), "\n")
}
```

```
## Column: Team_Index , Class: character
## Column: Country.abr , Class: character
## Column: Player , Class: character
## Column: FirstName , Class: character
## Column: LastName , Class: character
## Column: Position , Class: character
## Column: Height , Class: numeric
## Column: Weight , Class: numeric
## Column: BirthDate , Class: POSIXct POSIXt
## Column: College , Class: character
## Column: Address , Class: character
## Column: City , Class: character
## Column: State , Class: character
## Column: Zip , Class: numeric
## Column: County , Class: character
## Column: Latitude , Class: numeric
## Column: Longitude , Class: numeric
## Column: Country , Class: character
## Column: Country.abr3 , Class: character
## Column: Country.num , Class: integer
## Column: Cty.Latitude , Class: numeric
## Column: Cty.Longitude , Class: numeric
## Column: Team , Class: character
```

```r
for (col in names(NBA_Roster)) {
  if (is.numeric(NBA_Roster[[col]])) {
    cat("Column:", col, "\n")
    print(summary(NBA_Roster[[col]]))
  }
}
```

```
## Column: Height
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   69.00   77.00   79.00   78.74   81.00   88.00
## Column: Weight
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   160.0   200.0   215.0   215.4   230.0   290.0
## Column: Zip
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0   23284   40506   45003   72701   99258
## Column: Latitude
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   31.55   36.07   31.67   39.71   47.67
## Column: Longitude
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -123.28  -95.95  -84.50  -78.26  -78.67    0.00
## Column: Country.num
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    24.0   840.0   840.0   726.7   840.0   840.0
## Column: Cty.Latitude
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -41.00   38.00   38.00   37.23   38.00   64.00
## Column: Cty.Longitude
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -97.00  -97.00  -97.00  -74.94  -97.00  174.00
```

```r
for (col in names(NBA_Roster)) {
  cat("Column:", col, ", NA count:", sum(is.na(NBA_Roster[[col]])), "\n")
}
```

```
## Column: Team_Index , NA count: 0
## Column: Country.abr , NA count: 0
## Column: Player , NA count: 0
## Column: FirstName , NA count: 0
## Column: LastName , NA count: 0
## Column: Position , NA count: 0
## Column: Height , NA count: 0
## Column: Weight , NA count: 0
## Column: BirthDate , NA count: 0
## Column: College , NA count: 0
## Column: Address , NA count: 0
## Column: City , NA count: 0
## Column: State , NA count: 0
## Column: Zip , NA count: 0
## Column: County , NA count: 0
## Column: Latitude , NA count: 0
## Column: Longitude , NA count: 0
```

```
## Column: Country , NA count: 0
## Column: Country.abr3 , NA count: 0
## Column: Country.num , NA count: 0
## Column: Cty.Latitude , NA count: 0
## Column: Cty.Longitude , NA count: 0
## Column: Team , NA count: 0
```

```
for (col in names(NBA_Roster)) {
  if (is.numeric(NBA_Roster[[col]])) {
    non_numeric <- !grepl("^-?\\d+\\.?\\d*$", NBA_Roster[[col]])
    cat("Column:", col, ", Non-numeric count:", sum(non_numeric), "\n")
  }
}
```

```
## Column: Height , Non-numeric count: 0
## Column: Weight , Non-numeric count: 0
## Column: Zip , Non-numeric count: 0
## Column: Latitude , Non-numeric count: 0
## Column: Longitude , Non-numeric count: 0
## Column: Country.num , Non-numeric count: 0
## Column: Cty.Latitude , Non-numeric count: 0
## Column: Cty.Longitude , Non-numeric count: 0
```

```
for (col in names(NBA_Roster)) {
  na_rows <- which(is.na(NBA_Roster[[col]]))
  if (length(na_rows) > 0) {
    cat("Column:", col, ", NA rows:", na_rows, "\n")
  }
}

for (col in names(NBA_Roster)) {
  if (is.numeric(NBA_Roster[[col]])) {
    non_numeric_rows <- which(grepl("[^0-9.]", NBA_Roster[[col]]))
    if (length(non_numeric_rows) > 0) {
      cat("Column:", col, ", Non-numeric rows:", non_numeric_rows, "\n")
    }
  }
}
```

```
## Column: Longitude , Non-numeric rows: 1 7 8 9 10 11 12 13 14 15 16 17 18 19 20 22 23 24 25 26 27 28 1
## Column: Cty.Latitude , Non-numeric rows: 1 36 108 109 110 161 178 180 268 321 356 374
## Column: Cty.Longitude , Non-numeric rows: 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 22 24 25 26 27 1
```

```
for (col in names(NBA_Roster)) {
  if (is.numeric(NBA_Roster[[col]])) {
    non_numeric_rows <- which(!grepl("^-?\\d+\\.?\\d*$", NBA_Roster[[col]]))
    if (length(non_numeric_rows) > 0) {
      cat("Column:", col, "\n")
      print(NBA_Roster[non_numeric_rows, ])
    } else {
      cat("No non-numeric values found in column:", col, "\n")
    }
  }
}
```

```
## No non-numeric values found in column: Height
## No non-numeric values found in column: Weight
## No non-numeric values found in column: Zip
## No non-numeric values found in column: Latitude
## No non-numeric values found in column: Longitude
## No non-numeric values found in column: Country.num
## No non-numeric values found in column: Cty.Latitude
## No non-numeric values found in column: Cty.Longitude
```

6.1.9 It seem our NBA_Roster df is Clean.

6.2 Per Game

6.2.1 Build a function to run our data frames through. To check for `NA/NULL` values, non-numeric values and find the column / row the errors are in. In order to save space, you can find an `image of the results` run by the check code.

6.2.2 In this example we are going to use NBA_Per_Game.

6.2.3 The actual code is below, run on the cleaned df to show in use...

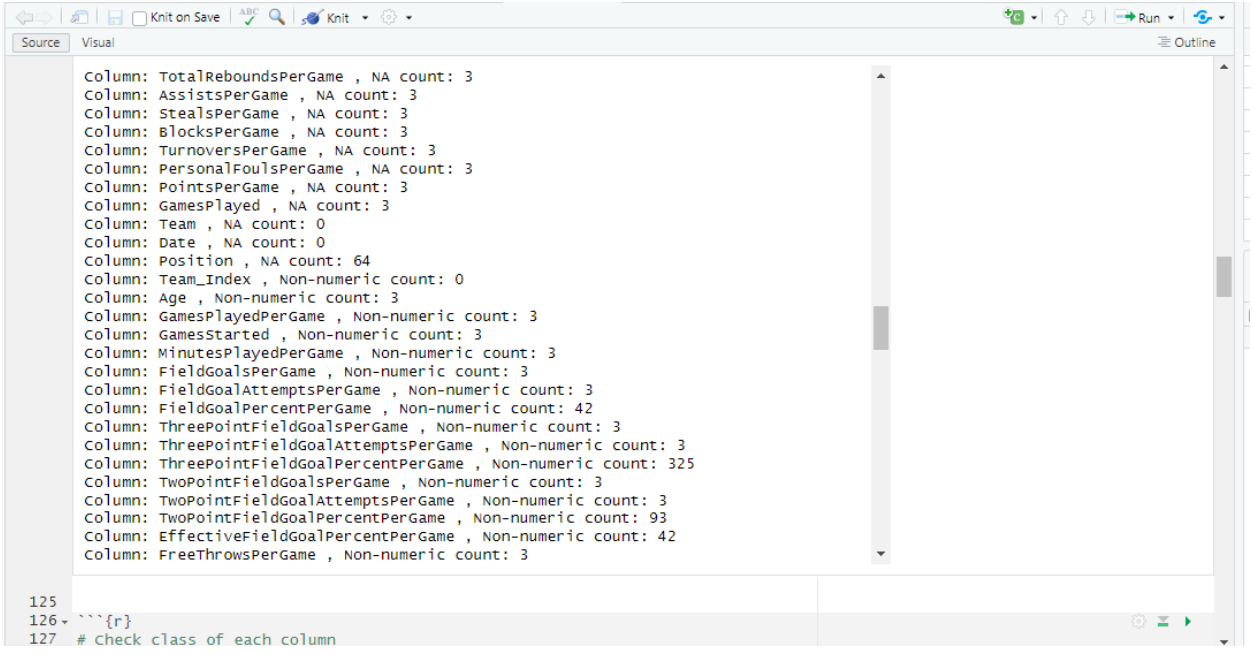


Figure 2: Per_Game Check before cleaning

6.2.4 We are going to produce a list of player names that have no position value and see if we can correct them.

6.2.5 First find rows with NA values in the Position column using the `subset` function

6.2.6 Extract names from the Player column into a new df "players_with_na_positions"

6.2.7 Print the names

```
na_positions <- subset(NBA_Per_Game, is.na(Position))

players_with_na_position <- na_positions$Player

print(players_with_na_position)
```

```
##  [1] "armoni brooks"         "théo maledon"          "frank ntilikina"
##  [4] "james bouknight"       "nathan mensah"         "ish smith"
##  [7] "terry taylor"          "dexter dennis"         "jaylen nowell"
## [10] "tosan evbuomwan"       "killian hayes"         "kevin knox"
## [13] "isaiah livers"         "joe harris"            "malcolm cazalon"
## [16] "cory joseph"           "joshua primo"          "d'moi hodge"
## [19] "mãozinha pereira"      "tosan evbuomwan"       "zavier simpson"
## [22] "dejon jarreau"         "wenyen gabriel"        "timmy allen"
## [25] "jaylen nowell"         "matthew hurt"          "jack white"
## [28] "shaquille harrison"    "dru smith"             "r.j. hampton"
## [31] "lindell wigginton"     "robin lopez"           "justin jackson"
## [34] "izaiah brockington"    "jalen crutcher"        "kaiser gates"
## [37] "ryan arcidiacono"      "dmytro skapintsev"     "d.j. wilson"
## [40] "filip petrušev"        "danny green"           "danuel house jr."
## [43] "furkan korkmaz"        "javonte smart"         "théo maledon"
## [46] "taze moore"            "filip petrušev"        "juan toscano-anderson"
## [49] "markquis nowell"       "kobi simmons"          "jahmi'us ramsey"
## [52] "ron harper jr."        "malik williams"        "jontay porter"
## [55] "otto porter jr."       "hamidou diallo"
```

6.2.8 Make a function to fix the POsitions column with NA values using a list we made from findin the players positions.

```
fix_na_positions <- function(df, player_list) {
  for (i in seq(1, length(player_list), by = 2)) {
    player <- player_list[i]
    position <- player_list[i + 1]
    df$Position[df$Player == player & is.na(df$Position)] <- position
  }
  return(df)
}
```

6.2.9 Compile the list.

```
player_list <- c("ersan ilyasova", "pf", "ersan ilyasova", "pf", "ersan ilyasova", "pf",
                 "ersan ilyasova", "pf", "ersan ilyasova", "pf", "armoni brooks", "sg",
                 "terry taylor", "pf", "ish smith", "pg", "théo maledon", "pg",
                 "nathan mensah", "c", "frank ntilikina", "sg", "james bouknight", "sg",
                 "dexter dennis", "sg", "killian hayes", "pg", "isaiah livers", "sf",
                 "kevin knox", "pf", "tosan evbuomwan", "sf", "jaylen nowell", "sg", "joe harris", "sg"
                 "malcolm cazalon", "sg", "cory joseph", "sg", "joshua primo", "sg",
                 "d'moi hodge", "sg", "timmy allen", "sf", "zavier simpson", "pg",
                 "mãozinha pereira", "sf", "jaylen nowell", "sg", "dejon jarreau", "sg",
                 "wenyen gabriel", "pf", "jack white", "sf", "matthew hurt", "pf",
                 "shaquille harrison", "sg", "dru smith", "sg", "r.j. hampton", "sg",
                 "robin lopez", "c", "lindell wigginton", "pg", "justin jackson", "pf",
                 "kaiser gates", "sf", "jalen crutcher", "pg", "izaiah brockington", "pg",
                 "ryan arcidiacono", "pg", "dmytro skapintsev", "c", "danuel house jr.", "sf",
                 "danny green", "sg", "ricky council iv", "sg", "furkan korkmaz", "sg",
                 "d.j. wilson", "pf", "filip petrušev", "c", "javonte smart", "pg",
                 "théo maledon", "pg", "taze moore", "sg", "juan toscano-anderson", "sf",
                 "filip petrušev", "c", "jahmi'us ramsey", "sg", "kobi simmons", "pg",
                 "otto porter jr.", "pf", "markquis nowell", "sg", "ron harper jr.", "pf", "malik willi
                 "hamidou diallo", "sg")
```

6.2.10 Then execute the function to assign players their positions.

```
NBA_Per_Game <- fix_na_positions(NBA_Per_Game, player_list)
```

6.2.11 Next we see that almost every numeric row has NA values in them.

6.2.12 Because these are sports statistics any NA value most likely means 0, so we are goin to replace the numeric valuse with 0.

```
NBA_Per_Game$Team_Index <- as.numeric(NBA_Per_Game$Team_Index)
NBA_Per_Game$FieldGoalPercentPerGame[is.na(NBA_Per_Game$FieldGoalPercentPerGame)] <- 0
NBA_Per_Game$ThreePointFieldGoalPercentPerGame[is.na(NBA_Per_Game$ThreePointFieldGoalPercentPerGame)] <-
NBA_Per_Game$TwoPointFieldGoalPercentPerGame[is.na(NBA_Per_Game$TwoPointFieldGoalPercentPerGame)] <- 0
NBA_Per_Game$EffectiveFieldGoalPercentPerGame[is.na(NBA_Per_Game$EffectiveFieldGoalPercentPerGame)] <- 
NBA_Per_Game$FreeThrowPercentPerGame[is.na(NBA_Per_Game$FreeThrowPercentPerGame)] <- 0
NBA_Per_Game$GamesPlayed[is.na(NBA_Per_Game$GamesPlayed)] <- 0
NBA_Per_Game$Age[is.na(NBA_Per_Game$Age)] <- 0
NBA_Per_Game$GamesPlayedPerGame[is.na(NBA_Per_Game$GamesPlayedPerGame)] <- 0
NBA_Per_Game$GamesStarted[is.na(NBA_Per_Game$GamesStarted)] <- 0
NBA_Per_Game$MinutesPlayedPerGame[is.na(NBA_Per_Game$MinutesPlayedPerGame)] <- 0
NBA_Per_Game$FieldGoalsPerGame[is.na(NBA_Per_Game$FieldGoalsPerGame)] <- 0
NBA_Per_Game$FieldGoalAttemptsPerGame[is.na(NBA_Per_Game$FieldGoalAttemptsPerGame)] <- 0
NBA_Per_Game$ThreePointFieldGoalsPerGame[is.na(NBA_Per_Game$ThreePointFieldGoalsPerGame)] <- 0
NBA_Per_Game$ThreePointFieldGoalAttemptsPerGame[is.na(NBA_Per_Game$ThreePointFieldGoalAttemptsPerGame)]
NBA_Per_Game$TwoPointFieldGoalsPerGame[is.na(NBA_Per_Game$TwoPointFieldGoalsPerGame)] <- 0
```

```
NBA_Per_Game$TwoPointFieldGoalAttemptsPerGame[is.na(NBA_Per_Game$TwoPointFieldGoalAttemptsPerGame)] <- 0
NBA_Per_Game$FreeThrowsPerGame[is.na(NBA_Per_Game$FreeThrowsPerGame)] <- 0
NBA_Per_Game$FreeThrowAttemptsPerGame[is.na(NBA_Per_Game$FreeThrowAttemptsPerGame)] <- 0
NBA_Per_Game$OffensiveReboundsPerGame[is.na(NBA_Per_Game$OffensiveReboundsPerGame)] <- 0
NBA_Per_Game$DefensiveReboundsPerGame[is.na(NBA_Per_Game$DefensiveReboundsPerGame)] <- 0
NBA_Per_Game$TotalReboundsPerGame[is.na(NBA_Per_Game$TotalReboundsPerGame)] <- 0
NBA_Per_Game$AssistsPerGame[is.na(NBA_Per_Game$AssistsPerGame)] <- 0
NBA_Per_Game$StealsPerGame[is.na(NBA_Per_Game$StealsPerGame)] <- 0
NBA_Per_Game$BlocksPerGame[is.na(NBA_Per_Game$BlocksPerGame)] <- 0
NBA_Per_Game$TurnoversPerGame[is.na(NBA_Per_Game$TurnoversPerGame)] <- 0
NBA_Per_Game$PersonalFoulsPerGame[is.na(NBA_Per_Game$PersonalFoulsPerGame)] <- 0
NBA_Per_Game$PointsPerGame[is.na(NBA_Per_Game$PointsPerGame)] <- 0
```

6.2.13 Let's re-run the check to make sure our df is now clean.

```
for (col in names(NBA_Per_Game)) {
  cat("Column:", col, ", Class:", class(NBA_Per_Game[[col]]), "\n")
}
```

```
## Column: Team_Index , Class: numeric
## Column: Player , Class: character
## Column: Age , Class: numeric
## Column: GamesPlayedPerGame , Class: numeric
## Column: GamesStarted , Class: numeric
## Column: MinutesPlayedPerGame , Class: numeric
## Column: FieldGoalsPerGame , Class: numeric
## Column: FieldGoalAttemptsPerGame , Class: numeric
## Column: FieldGoalPercentPerGame , Class: numeric
## Column: ThreePointFieldGoalsPerGame , Class: numeric
## Column: ThreePointFieldGoalAttemptsPerGame , Class: numeric
## Column: ThreePointFieldGoalPercentPerGame , Class: numeric
## Column: TwoPointFieldGoalsPerGame , Class: numeric
## Column: TwoPointFieldGoalAttemptsPerGame , Class: numeric
## Column: TwoPointFieldGoalPercentPerGame , Class: numeric
## Column: EffectiveFieldGoalPercentPerGame , Class: numeric
## Column: FreeThrowsPerGame , Class: numeric
## Column: FreeThrowAttemptsPerGame , Class: numeric
## Column: FreeThrowPercentPerGame , Class: numeric
## Column: OffensiveReboundsPerGame , Class: numeric
## Column: DefensiveReboundsPerGame , Class: numeric
## Column: TotalReboundsPerGame , Class: numeric
## Column: AssistsPerGame , Class: numeric
## Column: StealsPerGame , Class: numeric
## Column: BlocksPerGame , Class: numeric
## Column: TurnoversPerGame , Class: numeric
## Column: PersonalFoulsPerGame , Class: numeric
## Column: PointsPerGame , Class: numeric
## Column: GamesPlayed , Class: numeric
## Column: PER , Class: numeric
## Column: Position , Class: character
## Column: Team , Class: character
```

```r
for (col in names(NBA_Per_Game)) {
  if (is.numeric(NBA_Per_Game[[col]])) {
    cat("Column:", col, "\n")
    print(summary(NBA_Per_Game[[col]]))
  }
}
```

```
## Column: Team_Index
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    1.00    9.00   16.00   15.87   23.00   30.00 
## Column: Age
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   19.00   23.00   25.00   25.98   29.00   39.00 
## Column: GamesPlayedPerGame
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    1.00   16.00   39.00   40.18   65.00   82.00 
## Column: GamesStarted
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    0.00    0.00    5.00   18.72   30.00   82.00 
## Column: MinutesPlayedPerGame
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    0.50   10.20   17.20   18.32   26.90   37.80 
## Column: FieldGoalsPerGame
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   0.000   1.200   2.300   3.019   4.400  11.500 
## Column: FieldGoalAttemptsPerGame
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   0.000   2.900   5.000   6.506   9.100  23.600 
## Column: FieldGoalPercentPerGame
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  0.0000  0.4000  0.4480  0.4418  0.5000  0.8000 
## Column: ThreePointFieldGoalsPerGame
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  0.0000  0.2000  0.7000  0.9189  1.4000  4.8000 
## Column: ThreePointFieldGoalAttemptsPerGame
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   0.000   0.900   2.100   2.601   3.800  11.800 
## Column: ThreePointFieldGoalPercentPerGame
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  0.0000  0.2500  0.3360  0.2964  0.3840  1.0000 
## Column: TwoPointFieldGoalsPerGame
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##     0.0     0.7     1.5     2.1     2.9    11.0 
## Column: TwoPointFieldGoalAttemptsPerGame
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   0.000   1.400   2.800   3.904   5.300  18.300 
## Column: TwoPointFieldGoalPercentPerGame
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  0.0000  0.4740  0.5300  0.5098  0.5820  1.0000 
## Column: EffectiveFieldGoalPercentPerGame
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  0.0000  0.4820  0.5290  0.5089  0.5760  0.9170 
## Column: FreeThrowsPerGame
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.300   0.800   1.196   1.600  10.200
## Column: FreeThrowAttemptsPerGame
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    0.50    1.00    1.54    2.10   11.60
## Column: FreeThrowPercentPerGame
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.6430  0.7590  0.6822  0.8330  1.0000
## Column: OffensiveReboundsPerGame
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.3000  0.6000  0.8297  1.1000  4.6000
## Column: DefensiveReboundsPerGame
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.100   2.100   2.454   3.400  10.100
## Column: TotalReboundsPerGame
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.500   2.800   3.278   4.400  13.700
## Column: AssistsPerGame
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.700   1.300   1.953   2.600  10.900
## Column: StealsPerGame
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.300   0.500   0.588   0.900   2.100
## Column: BlocksPerGame
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.1000  0.3000  0.3889  0.5000  3.6000
## Column: TurnoversPerGame
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.4000  0.7000  0.9557  1.3000  4.4000
## Column: PersonalFoulsPerGame
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.900   1.500   1.462   2.000   3.600
## Column: PointsPerGame
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   3.300   6.100   8.149  11.500  34.700
## Column: GamesPlayed
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00   16.00   39.00   40.18   65.00   82.00
## Column: PER
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -3.0000  0.1364  0.2197  0.3530  0.3680  6.0000
```

```r
for (col in names(NBA_Per_Game)) {
  cat("Column:", col, ", NA count:", sum(is.na(NBA_Per_Game[[col]])), "\n")
}
```

```
## Column: Team_Index , NA count: 0
## Column: Player , NA count: 0
## Column: Age , NA count: 0
## Column: GamesPlayedPerGame , NA count: 0
## Column: GamesStarted , NA count: 0
## Column: MinutesPlayedPerGame , NA count: 0
## Column: FieldGoalsPerGame , NA count: 0
## Column: FieldGoalAttemptsPerGame , NA count: 0
```

```
## Column: FieldGoalPercentPerGame , NA count: 0
## Column: ThreePointFieldGoalsPerGame , NA count: 0
## Column: ThreePointFieldGoalAttemptsPerGame , NA count: 0
## Column: ThreePointFieldGoalPercentPerGame , NA count: 0
## Column: TwoPointFieldGoalsPerGame , NA count: 0
## Column: TwoPointFieldGoalAttemptsPerGame , NA count: 0
## Column: TwoPointFieldGoalPercentPerGame , NA count: 0
## Column: EffectiveFieldGoalPercentPerGame , NA count: 0
## Column: FreeThrowsPerGame , NA count: 0
## Column: FreeThrowAttemptsPerGame , NA count: 0
## Column: FreeThrowPercentPerGame , NA count: 0
## Column: OffensiveReboundsPerGame , NA count: 0
## Column: DefensiveReboundsPerGame , NA count: 0
## Column: TotalReboundsPerGame , NA count: 0
## Column: AssistsPerGame , NA count: 0
## Column: StealsPerGame , NA count: 0
## Column: BlocksPerGame , NA count: 0
## Column: TurnoversPerGame , NA count: 0
## Column: PersonalFoulsPerGame , NA count: 0
## Column: PointsPerGame , NA count: 0
## Column: GamesPlayed , NA count: 0
## Column: PER , NA count: 0
## Column: Position , NA count: 0
## Column: Team , NA count: 0
```

```
for (col in names(NBA_Per_Game)) {
  if (is.numeric(NBA_Per_Game[[col]])) {
    non_numeric <- !grepl("^-?\\d+\\.?\\d*$", NBA_Per_Game[[col]])
    cat("Column:", col, ", Non-numeric count:", sum(non_numeric), "\n")
  }
}
```

```
## Column: Team_Index , Non-numeric count: 0
## Column: Age , Non-numeric count: 0
## Column: GamesPlayedPerGame , Non-numeric count: 0
## Column: GamesStarted , Non-numeric count: 0
## Column: MinutesPlayedPerGame , Non-numeric count: 0
## Column: FieldGoalsPerGame , Non-numeric count: 0
## Column: FieldGoalAttemptsPerGame , Non-numeric count: 0
## Column: FieldGoalPercentPerGame , Non-numeric count: 0
## Column: ThreePointFieldGoalsPerGame , Non-numeric count: 0
## Column: ThreePointFieldGoalAttemptsPerGame , Non-numeric count: 0
## Column: ThreePointFieldGoalPercentPerGame , Non-numeric count: 0
## Column: TwoPointFieldGoalsPerGame , Non-numeric count: 0
## Column: TwoPointFieldGoalAttemptsPerGame , Non-numeric count: 0
## Column: TwoPointFieldGoalPercentPerGame , Non-numeric count: 0
## Column: EffectiveFieldGoalPercentPerGame , Non-numeric count: 0
## Column: FreeThrowsPerGame , Non-numeric count: 0
## Column: FreeThrowAttemptsPerGame , Non-numeric count: 0
## Column: FreeThrowPercentPerGame , Non-numeric count: 0
## Column: OffensiveReboundsPerGame , Non-numeric count: 0
## Column: DefensiveReboundsPerGame , Non-numeric count: 0
## Column: TotalReboundsPerGame , Non-numeric count: 0
## Column: AssistsPerGame , Non-numeric count: 0
```

```
## Column: StealsPerGame , Non-numeric count: 0
## Column: BlocksPerGame , Non-numeric count: 0
## Column: TurnoversPerGame , Non-numeric count: 0
## Column: PersonalFoulsPerGame , Non-numeric count: 0
## Column: PointsPerGame , Non-numeric count: 0
## Column: GamesPlayed , Non-numeric count: 0
## Column: PER , Non-numeric count: 0
```

```
for (col in names(NBA_Per_Game)) {
  na_rows <- which(is.na(NBA_Per_Game[[col]]))
  if (length(na_rows) > 0) {
    cat("Column:", col, ", NA rows:", na_rows, "\n")
  }
}

for (col in names(NBA_Per_Game)) {
  if (is.numeric(NBA_Per_Game[[col]])) {
    non_numeric_rows <- which(grepl("[^0-9.]", NBA_Per_Game[[col]]))
    if (length(non_numeric_rows) > 0) {
      cat("Column:", col, ", Non-numeric rows:", non_numeric_rows, "\n")
    }
  }
}
```

```
## Column: PER , Non-numeric rows: 103 121 248 269 290 344 403 404 430 498
```

```
for (col in names(NBA_Per_Game)) {
  if (is.numeric(NBA_Per_Game[[col]])) {
    non_numeric_rows <- which(!grepl("^-?\\d+\\.?\\d*$", NBA_Per_Game[[col]]))
    if (length(non_numeric_rows) > 0) {
      cat("Column:", col, "\n")
      print(NBA_Per_Game[non_numeric_rows, ])
    } else {
      cat("No non-numeric values found in column:", col, "\n")
    }
  }
}
```

```
## No non-numeric values found in column: Team_Index
## No non-numeric values found in column: Age
## No non-numeric values found in column: GamesPlayedPerGame
## No non-numeric values found in column: GamesStarted
## No non-numeric values found in column: MinutesPlayedPerGame
## No non-numeric values found in column: FieldGoalsPerGame
## No non-numeric values found in column: FieldGoalAttemptsPerGame
## No non-numeric values found in column: FieldGoalPercentPerGame
## No non-numeric values found in column: ThreePointFieldGoalsPerGame
## No non-numeric values found in column: ThreePointFieldGoalAttemptsPerGame
## No non-numeric values found in column: ThreePointFieldGoalPercentPerGame
## No non-numeric values found in column: TwoPointFieldGoalsPerGame
## No non-numeric values found in column: TwoPointFieldGoalAttemptsPerGame
## No non-numeric values found in column: TwoPointFieldGoalPercentPerGame
## No non-numeric values found in column: EffectiveFieldGoalPercentPerGame
```

```
## No non-numeric values found in column: FreeThrowsPerGame
## No non-numeric values found in column: FreeThrowAttemptsPerGame
## No non-numeric values found in column: FreeThrowPercentPerGame
## No non-numeric values found in column: OffensiveReboundsPerGame
## No non-numeric values found in column: DefensiveReboundsPerGame
## No non-numeric values found in column: TotalReboundsPerGame
## No non-numeric values found in column: AssistsPerGame
## No non-numeric values found in column: StealsPerGame
## No non-numeric values found in column: BlocksPerGame
## No non-numeric values found in column: TurnoversPerGame
## No non-numeric values found in column: PersonalFoulsPerGame
## No non-numeric values found in column: PointsPerGame
## No non-numeric values found in column: GamesPlayed
## No non-numeric values found in column: PER
```

6.2.14 Now that we can confirm that data frame is now clean we can move onto the next.

6.2.15 Continue this process for NBA_Totals, NBA_Starting_Lineups, NBA_Game_Log and NBA_Results...

6.3 Totals

6.3.1 We have to assign missing positions to player in our `Totals` df also, so let's go ahead and do that now.

6.3.2 First, make sure it is the same list of players.

```
na_positions <- subset(NBA_Totals, is.na(Position))

players_with_na_position <- na_positions$Player

print(players_with_na_position)
```

```
##  [1] "armoni brooks"         "ish smith"             "nathan mensah"
##  [4] "théo maledon"          "james bouknight"       "frank ntilikina"
##  [7] "terry taylor"          "dexter dennis"         "killian hayes"
## [10] "kevin knox"            "isaiah livers"         "tosan evbuomwan"
## [13] "joe harris"            "jaylen nowell"         "malcolm cazalon"
## [16] "cory joseph"           "joshua primo"          "d'moi hodge"
## [19] "zavier simpson"        "jaylen nowell"         "dejon jarreau"
## [22] "timmy allen"           "mãozinha pereira"      "matthew hurt"
## [25] "wenyen gabriel"        "tosan evbuomwan"       "jack white"
## [28] "shaquille harrison"    "dru smith"             "r.j. hampton"
## [31] "robin lopez"           "lindell wigginton"     "justin jackson"
## [34] "kaiser gates"          "jalen crutcher"        "izaiah brockington"
## [37] "ryan arcidiacono"      "dmytro skapintsev"     "danuel house jr."
## [40] "furkan korkmaz"        "danny green"           "d.j. wilson"
## [43] "filip petrušev"        "javonte smart"         "théo maledon"
## [46] "taze moore"            "juan toscano-anderson" "filip petrušev"
## [49] "jontay porter"         "otto porter jr."       "jahmi'us ramsey"
## [52] "malik williams"        "kobi simmons"          "markquis nowell"
## [55] "ron harper jr."        "hamidou diallo"
```

6.3.3 Then execute the function to assign players their positions.

```
NBA_Totals <- fix_na_positions(NBA_Totals, player_list)
```

6.3.4 Let's add 0's for any NA's in our numeric `Totals` df.

```
NBA_Totals$Team_Index <- as.numeric(NBA_Totals$Team_Index)
NBA_Totals$TotalGamesPlayed[is.na(NBA_Totals$TotalGamesPlayed)] <- 0
NBA_Totals$TotalGamesStarted[is.na(NBA_Totals$TotalGamesStarted)] <- 0
NBA_Totals$TotalMinutesPlayed[is.na(NBA_Totals$TotalMinutesPlayed)] <- 0
NBA_Totals$TotalFieldGoalsPerGame[is.na(NBA_Totals$TotalFieldGoalsPerGame)] <- 0
NBA_Totals$TotalFieldGoalAttempts[is.na(NBA_Totals$TotalFieldGoalAttempts)] <- 0
NBA_Totals$TotalFieldGoalPercent[is.na(NBA_Totals$TotalFieldGoalPercent)] <- 0
NBA_Totals$TotalThreePointFieldGoalsPerGame[is.na(NBA_Totals$TotalThreePointFieldGoalsPerGame)] <- 0
NBA_Totals$TotalThreePointFieldGoalAttempts[is.na(NBA_Totals$TotalThreePointFieldGoalAttempts)] <- 0
NBA_Totals$TotalThreePointFieldGoalPercent[is.na(NBA_Totals$TotalThreePointFieldGoalPercent)] <- 0
NBA_Totals$TotalTwoPointFieldGoalsPerGame[is.na(NBA_Totals$TotalTwoPointFieldGoalsPerGame)] <- 0
NBA_Totals$TotalTwoPointFieldGoalAttempts[is.na(NBA_Totals$TotalTwoPointFieldGoalAttempts)] <- 0
NBA_Totals$TotalTwoPointFieldGoalPercent[is.na(NBA_Totals$TotalTwoPointFieldGoalPercent)] <- 0
NBA_Totals$TotalEffectiveFieldGoalPercent[is.na(NBA_Totals$TotalEffectiveFieldGoalPercent)] <- 0
NBA_Totals$TotalFreeThrows[is.na(NBA_Totals$TotalFreeThrows)] <- 0
NBA_Totals$TotalFreeThrowAttempts[is.na(NBA_Totals$TotalFreeThrowAttempts)] <- 0
NBA_Totals$TotalFreeThrowPercent[is.na(NBA_Totals$TotalFreeThrowPercent)] <- 0
NBA_Totals$TotalOffensiveRebounds[is.na(NBA_Totals$TotalOffensiveRebounds)] <- 0
NBA_Totals$TotalDefensiveRebounds[is.na(NBA_Totals$TotalDefensiveRebounds)] <- 0
NBA_Totals$TotalRebounds[is.na(NBA_Totals$TotalRebounds)] <- 0
NBA_Totals$TotalAssists[is.na(NBA_Totals$TotalAssists)] <- 0
NBA_Totals$TotalSteals[is.na(NBA_Totals$TotalSteals)] <- 0
NBA_Totals$TotalBlocks[is.na(NBA_Totals$TotalBlocks)] <- 0
NBA_Totals$TotalTurnovers[is.na(NBA_Totals$TotalTurnovers)] <- 0
NBA_Totals$TotalPersonalFouls[is.na(NBA_Totals$TotalPersonalFouls)] <- 0
NBA_Totals$TotalPoints[is.na(NBA_Totals$TotalPoints)] <- 0

colnames(NBA_Totals)[9] <- "TotalFieldGoals"
colnames(NBA_Per_Game)[12] <- "TotalThreePointFieldGoals"
colnames(NBA_Per_Game)[15] <- "TotalTwoPointFieldGoals"
```

6.3.5 Then Re-run the check

```
for (col in names(NBA_Totals)) {
  cat("Column:", col, ", Class:", class(NBA_Totals[[col]]), "\n")
}
```

```
## Column: Team_Index , Class: numeric
## Column: Player , Class: character
## Column: Age , Class: numeric
## Column: TotalGamesPlayed , Class: numeric
## Column: TotalGamesStarted , Class: numeric
```

```
## Column: TotalMinutesPlayed , Class: numeric
## Column: TotalFieldGoalsPerGame , Class: numeric
## Column: TotalFieldGoalAttempts , Class: numeric
## Column: TotalFieldGoals , Class: numeric
## Column: TotalThreePointFieldGoalsPerGame , Class: numeric
## Column: TotalThreePointFieldGoalAttempts , Class: numeric
## Column: TotalThreePointFieldGoalPercent , Class: numeric
## Column: TotalTwoPointFieldGoalsPerGame , Class: numeric
## Column: TotalTwoPointFieldGoalAttempts , Class: numeric
## Column: TotalTwoPointFieldGoalPercent , Class: numeric
## Column: TotalEffectiveFieldGoalPercent , Class: numeric
## Column: TotalFreeThrows , Class: numeric
## Column: TotalFreeThrowAttempts , Class: numeric
## Column: TotalFreeThrowPercent , Class: numeric
## Column: TotalOffensiveRebounds , Class: numeric
## Column: TotalDefensiveRebounds , Class: numeric
## Column: TotalRebounds , Class: numeric
## Column: TotalAssists , Class: numeric
## Column: TotalSteals , Class: numeric
## Column: TotalBlocks , Class: numeric
## Column: TotalTurnovers , Class: numeric
## Column: TotalPersonalFouls , Class: numeric
## Column: TotalPoints , Class: numeric
## Column: Team , Class: character
## Column: Position , Class: character
```

```
for (col in names(NBA_Totals)) {
  if (is.numeric(NBA_Totals[[col]])) {
    cat("Column:", col, "\n")
    print(summary(NBA_Totals[[col]]))
  }
}
```

```
## Column: Team_Index
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    9.00   16.00   15.87   23.00   30.00
## Column: Age
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   19.00   23.00   25.00   25.98   29.00   39.00
## Column: TotalGamesPlayed
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00   16.00   39.00   40.18   65.00   82.00
## Column: TotalGamesStarted
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    0.00    5.00   18.72   30.00   82.00
## Column: TotalMinutesPlayed
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0   142.0   654.0   903.7  1529.0  2989.0
## Column: TotalFieldGoalsPerGame
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0    18.0    88.0   157.9   239.0   837.0
## Column: TotalFieldGoalAttempts
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0    41.0   195.0   332.9   509.0  1652.0
```

```
## Column: TotalFieldGoals
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.4000  0.4480  0.4418  0.5000  0.8000
## Column: TotalThreePointFieldGoalsPerGame
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    2.00   21.00   48.07   77.00  357.00
## Column: TotalThreePointFieldGoalAttempts
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0     9.0    64.0   131.4   210.0   876.0
## Column: TotalThreePointFieldGoalPercent
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.2500  0.3360  0.2964  0.3840  1.0000
## Column: TotalTwoPointFieldGoalsPerGame
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0    11.0    54.0   109.8   151.0   803.0
## Column: TotalTwoPointFieldGoalAttempts
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0    23.0   105.0   201.4   279.0  1245.0
## Column: TotalTwoPointFieldGoalPercent
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.4740  0.5300  0.5098  0.5820  1.0000
## Column: TotalEffectiveFieldGoalPercent
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.4820  0.5290  0.5089  0.5760  0.9170
## Column: TotalFreeThrows
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    5.00   26.00   63.78   80.00  567.00
## Column: TotalFreeThrowAttempts
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    8.00   36.00   81.32  106.00  782.00
## Column: TotalFreeThrowPercent
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.6430  0.7590  0.6822  0.8330  1.0000
## Column: TotalOffensiveRebounds
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    6.00   22.00   39.52   55.00  335.00
## Column: TotalDefensiveRebounds
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0    17.0    80.0   123.5   187.0   826.0
## Column: TotalRebounds
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0      24     110     163     244    1120
## Column: TotalAssists
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   10.00   51.00   99.86  130.00  752.00
## Column: TotalSteals
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    5.00   20.00   27.99   43.00  150.00
## Column: TotalBlocks
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    2.00    9.00   19.25   28.00  254.00
## Column: TotalTurnovers
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0     7.0    29.0    48.3    69.0   282.0
```

```
## Column: TotalPersonalFouls
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   13.00   54.00   70.13  113.00  254.00
## Column: TotalPoints
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0    47.0   241.0   427.6   640.0  2370.0
```

```
for (col in names(NBA_Totals)) {
  cat("Column:", col, ", NA count:", sum(is.na(NBA_Totals[[col]])), "\n")
}
```

```
## Column: Team_Index , NA count: 0
## Column: Player , NA count: 0
## Column: Age , NA count: 0
## Column: TotalGamesPlayed , NA count: 0
## Column: TotalGamesStarted , NA count: 0
## Column: TotalMinutesPlayed , NA count: 0
## Column: TotalFieldGoalsPerGame , NA count: 0
## Column: TotalFieldGoalAttempts , NA count: 0
## Column: TotalFieldGoals , NA count: 0
## Column: TotalThreePointFieldGoalsPerGame , NA count: 0
## Column: TotalThreePointFieldGoalAttempts , NA count: 0
## Column: TotalThreePointFieldGoalPercent , NA count: 0
## Column: TotalTwoPointFieldGoalsPerGame , NA count: 0
## Column: TotalTwoPointFieldGoalAttempts , NA count: 0
## Column: TotalTwoPointFieldGoalPercent , NA count: 0
## Column: TotalEffectiveFieldGoalPercent , NA count: 0
## Column: TotalFreeThrows , NA count: 0
## Column: TotalFreeThrowAttempts , NA count: 0
## Column: TotalFreeThrowPercent , NA count: 0
## Column: TotalOffensiveRebounds , NA count: 0
## Column: TotalDefensiveRebounds , NA count: 0
## Column: TotalRebounds , NA count: 0
## Column: TotalAssists , NA count: 0
## Column: TotalSteals , NA count: 0
## Column: TotalBlocks , NA count: 0
## Column: TotalTurnovers , NA count: 0
## Column: TotalPersonalFouls , NA count: 0
## Column: TotalPoints , NA count: 0
## Column: Team , NA count: 0
## Column: Position , NA count: 0
```

```
for (col in names(NBA_Totals)) {
  if (is.numeric(NBA_Totals[[col]])) {
    non_numeric <- !grepl("^-?\\d+\\.?\\d*$", NBA_Totals[[col]])
    cat("Column:", col, ", Non-numeric count:", sum(non_numeric), "\n")
  }
}
```

```
## Column: Team_Index , Non-numeric count: 0
## Column: Age , Non-numeric count: 0
## Column: TotalGamesPlayed , Non-numeric count: 0
## Column: TotalGamesStarted , Non-numeric count: 0
```

```
## Column: TotalMinutesPlayed , Non-numeric count: 0
## Column: TotalFieldGoalsPerGame , Non-numeric count: 0
## Column: TotalFieldGoalAttempts , Non-numeric count: 0
## Column: TotalFieldGoals , Non-numeric count: 0
## Column: TotalThreePointFieldGoalsPerGame , Non-numeric count: 0
## Column: TotalThreePointFieldGoalAttempts , Non-numeric count: 0
## Column: TotalThreePointFieldGoalPercent , Non-numeric count: 0
## Column: TotalTwoPointFieldGoalsPerGame , Non-numeric count: 0
## Column: TotalTwoPointFieldGoalAttempts , Non-numeric count: 0
## Column: TotalTwoPointFieldGoalPercent , Non-numeric count: 0
## Column: TotalEffectiveFieldGoalPercent , Non-numeric count: 0
## Column: TotalFreeThrows , Non-numeric count: 0
## Column: TotalFreeThrowAttempts , Non-numeric count: 0
## Column: TotalFreeThrowPercent , Non-numeric count: 0
## Column: TotalOffensiveRebounds , Non-numeric count: 0
## Column: TotalDefensiveRebounds , Non-numeric count: 0
## Column: TotalRebounds , Non-numeric count: 0
## Column: TotalAssists , Non-numeric count: 0
## Column: TotalSteals , Non-numeric count: 0
## Column: TotalBlocks , Non-numeric count: 0
## Column: TotalTurnovers , Non-numeric count: 0
## Column: TotalPersonalFouls , Non-numeric count: 0
## Column: TotalPoints , Non-numeric count: 0
```

```
for (col in names(NBA_Totals)) {
  na_rows <- which(is.na(NBA_Totals[[col]]))
  if (length(na_rows) > 0) {
    cat("Column:", col, ", NA rows:", na_rows, "\n")
  }
}

for (col in names(NBA_Totals)) {
  if (is.numeric(NBA_Totals[[col]])) {
    non_numeric_rows <- which(grepl("[^0-9.]", NBA_Totals[[col]]))
    if (length(non_numeric_rows) > 0) {
      cat("Column:", col, ", Non-numeric rows:", non_numeric_rows, "\n")
    }
  }
}

for (col in names(NBA_Totals)) {
  if (is.numeric(NBA_Totals[[col]])) {
    non_numeric_rows <- which(!grepl("^-?\\d+\\.?\\d*$", NBA_Totals[[col]]))
    if (length(non_numeric_rows) > 0) {
      cat("Column:", col, "\n")
      print(NBA_Totals[non_numeric_rows, ])
    } else {
      cat("No non-numeric values found in column:", col, "\n")
    }
  }
}
```

```
## No non-numeric values found in column: Team_Index
## No non-numeric values found in column: Age
```

```
## No non-numeric values found in column: TotalGamesPlayed
## No non-numeric values found in column: TotalGamesStarted
## No non-numeric values found in column: TotalMinutesPlayed
## No non-numeric values found in column: TotalFieldGoalsPerGame
## No non-numeric values found in column: TotalFieldGoalAttempts
## No non-numeric values found in column: TotalFieldGoals
## No non-numeric values found in column: TotalThreePointFieldGoalsPerGame
## No non-numeric values found in column: TotalThreePointFieldGoalAttempts
## No non-numeric values found in column: TotalThreePointFieldGoalPercent
## No non-numeric values found in column: TotalTwoPointFieldGoalsPerGame
## No non-numeric values found in column: TotalTwoPointFieldGoalAttempts
## No non-numeric values found in column: TotalTwoPointFieldGoalPercent
## No non-numeric values found in column: TotalEffectiveFieldGoalPercent
## No non-numeric values found in column: TotalFreeThrows
## No non-numeric values found in column: TotalFreeThrowAttempts
## No non-numeric values found in column: TotalFreeThrowPercent
## No non-numeric values found in column: TotalOffensiveRebounds
## No non-numeric values found in column: TotalDefensiveRebounds
## No non-numeric values found in column: TotalRebounds
## No non-numeric values found in column: TotalAssists
## No non-numeric values found in column: TotalSteals
## No non-numeric values found in column: TotalBlocks
## No non-numeric values found in column: TotalTurnovers
## No non-numeric values found in column: TotalPersonalFouls
## No non-numeric values found in column: TotalPoints
```

6.3.6 The dataframe is now clean. On to the next

6.4 Starting_Lineups

6.4.1 Our Team_Index column should be numeric. Change that.

```
NBA_Starting_Lineups$Team_Index <- as.numeric(NBA_Starting_Lineups$Team_Index)
```

6.4.2 Run the check

```
for (col in names(NBA_Starting_Lineups)) {
  cat("Column:", col, ", Class:", class(NBA_Starting_Lineups[[col]]), "\n")
}
```

```
## Column: Team_Index , Class: numeric
## Column: Date , Class: Date
## Column: HomeAway , Class: logical
## Column: Opponent , Class: character
## Column: WinLoss , Class: logical
## Column: Overtime , Class: logical
## Column: TeamPoints , Class: numeric
## Column: OpponentPoints , Class: numeric
## Column: Wins , Class: numeric
```

```
## Column: Losses , Class: numeric
## Column: StartingLineup , Class: character
## Column: FirstName , Class: character
## Column: LastName , Class: character
## Column: Team , Class: character
```

```
for (col in names(NBA_Starting_Lineups)) {
  if (is.numeric(NBA_Starting_Lineups[[col]])) {
    cat("Column:", col, "\n")
    print(summary(NBA_Starting_Lineups[[col]]))
  }
}
```

```
## Column: Team_Index
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0     8.0    15.5    15.5    23.0    30.0
## Column: TeamPoints
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    73.0   105.0   114.0   114.2   123.0   157.0
## Column: OpponentPoints
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    73.0   105.0   114.0   114.2   123.0   157.0
## Column: Wins
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    9.00   18.00   20.75   31.00   64.00
## Column: Losses
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    9.00   19.00   20.75   29.00   68.00
```

```
for (col in names(NBA_Starting_Lineups)) {
  cat("Column:", col, ", NA count:", sum(is.na(NBA_Starting_Lineups[[col]])), "\n")
}
```

```
## Column: Team_Index , NA count: 0
## Column: Date , NA count: 0
## Column: HomeAway , NA count: 0
## Column: Opponent , NA count: 0
## Column: WinLoss , NA count: 0
## Column: Overtime , NA count: 0
## Column: TeamPoints , NA count: 0
## Column: OpponentPoints , NA count: 0
## Column: Wins , NA count: 0
## Column: Losses , NA count: 0
## Column: StartingLineup , NA count: 0
## Column: FirstName , NA count: 0
## Column: LastName , NA count: 0
## Column: Team , NA count: 0
```

```
for (col in names(NBA_Starting_Lineups)) {
  if (is.numeric(NBA_Starting_Lineups[[col]])) {
    non_numeric <- !grepl("^-?\\d+\\.?\\d*$", NBA_Starting_Lineups[[col]])
    cat("Column:", col, ", Non-numeric count:", sum(non_numeric), "\n")
  }
}
```

```
## Column: Team_Index , Non-numeric count: 0
## Column: TeamPoints , Non-numeric count: 0
## Column: OpponentPoints , Non-numeric count: 0
## Column: Wins , Non-numeric count: 0
## Column: Losses , Non-numeric count: 0
```

```
for (col in names(NBA_Starting_Lineups)) {
  na_rows <- which(is.na(NBA_Starting_Lineups[[col]]))
  if (length(na_rows) > 0) {
    cat("Column:", col, ", NA rows:", na_rows, "\n")
  }
}

for (col in names(NBA_Starting_Lineups)) {
  if (is.numeric(NBA_Starting_Lineups[[col]])) {
    non_numeric_rows <- which(grepl("[^0-9.]", NBA_Starting_Lineups[[col]]))
    if (length(non_numeric_rows) > 0) {
      cat("Column:", col, ", Non-numeric rows:", non_numeric_rows, "\n")
    }
  }
}

for (col in names(NBA_Starting_Lineups)) {
  if (is.numeric(NBA_Starting_Lineups[[col]])) {
    non_numeric_rows <- which(!grepl("^-?\\d+\\.?\\d*$", NBA_Starting_Lineups[[col]]))
    if (length(non_numeric_rows) > 0) {
      cat("Column:", col, "\n")
      print(NBA_Starting_Lineups[non_numeric_rows, ])
    } else {
      cat("No non-numeric values found in column:", col, "\n")
    }
  }
}
```

```
## No non-numeric values found in column: Team_Index
## No non-numeric values found in column: TeamPoints
## No non-numeric values found in column: OpponentPoints
## No non-numeric values found in column: Wins
## No non-numeric values found in column: Losses
```

6.4.3 This df seems to be clean

6.5 Game Log

6.5.1 There was a first time ever in a basketball game the Boston Celtics did not shoot a single free throw the entire game.

6.5.2 Because of this we can safely assume the NA values to be 0 for no FreeThrowPercentage or OppFreeThrowPercentage

```
NBA_Game_Log$Team_Index <- as.numeric(NBA_Game_Log$Team_Index)
NBA_Game_Log$FreeThrowPercent[is.na(NBA_Game_Log$FreeThrowPercent)] <- 0
NBA_Game_Log$OppFreeThrowPercent[is.na(NBA_Game_Log$OppFreeThrowPercent)] <- 0
```

6.5.2.1 So lets put 0's in their place

6.5.3 Run the Check

```
for (col in names(NBA_Game_Log)) {
  cat("Column:", col, ", Class:", class(NBA_Game_Log[[col]]), "\n")
}
```

```
## Column: Team_Index , Class: numeric
## Column: Date , Class: Date
## Column: Opp , Class: character
## Column: WinLoss , Class: logical
## Column: Points , Class: numeric
## Column: OppPoints , Class: numeric
## Column: FieldGoals , Class: numeric
## Column: FieldGoalAttempts , Class: numeric
## Column: FieldGoalPercent , Class: numeric
## Column: ThreePoints , Class: numeric
## Column: ThreePointAttempts , Class: numeric
## Column: ThreePointPercent , Class: numeric
## Column: FreeThrows , Class: numeric
## Column: FreeThrowAttempts , Class: numeric
## Column: FreeThrowPercent , Class: numeric
## Column: OffensiveRebounds , Class: numeric
## Column: TotalRebounds , Class: numeric
## Column: Assists , Class: numeric
## Column: Steals , Class: numeric
## Column: Blocks , Class: numeric
## Column: Turnovers , Class: numeric
## Column: PersonalFouls , Class: numeric
## Column: OppFieldGoals , Class: numeric
## Column: OppFieldGoalAttempts , Class: numeric
## Column: OppFieldGoalPercent , Class: numeric
## Column: OppThreePoints , Class: numeric
## Column: OppThreePointAttempts , Class: numeric
## Column: OppThreePointPercent , Class: numeric
## Column: OppFreeThrows , Class: numeric
## Column: OppFreeThrowAttempts , Class: numeric
## Column: OppFreeThrowPercent , Class: numeric
## Column: OppOffensiveRebounds , Class: numeric
## Column: OppTotalRebounds , Class: numeric
## Column: OppAssists , Class: numeric
## Column: OppSteals , Class: numeric
## Column: OppBlocks , Class: numeric
## Column: OppTurnovers , Class: numeric
```

```
## Column: OppPersonalFouls , Class: numeric
## Column: Opponent , Class: character
## Column: OpponentWins , Class: integer
## Column: OpponentLosses , Class: integer
## Column: OpponentStrength , Class: character
## Column: Team , Class: character
## Column: TeamWinLoss , Class: character
```

```
for (col in names(NBA_Game_Log)) {
  if (is.numeric(NBA_Game_Log[[col]])) {
    cat("Column:", col, "\n")
    print(summary(NBA_Game_Log[[col]]))
  }
}
```

```
## Column: Team_Index
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0     8.0    15.5    15.5    23.0    30.0
## Column: Points
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    73.0   105.0   114.0   114.2   123.0   157.0
## Column: OppPoints
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    73.0   105.0   114.0   114.2   123.0   157.0
## Column: FieldGoals
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   26.00   38.00   42.00   42.17   46.00   65.00
## Column: FieldGoalAttempts
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    67.0    84.0    89.0    88.9    93.0   119.0
## Column: FieldGoalPercent
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2770  0.4380  0.4750  0.4752  0.5120  0.6710
## Column: ThreePoints
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00   10.00   13.00   12.84   15.00   27.00
## Column: ThreePointAttempts
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    12.0    30.0    35.0    35.1    39.0    63.0
## Column: ThreePointPercent
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0690  0.3100  0.3655  0.3649  0.4170  0.6450
## Column: FreeThrows
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   13.00   17.00   17.03   21.00   44.00
## Column: FreeThrowAttempts
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   17.00   21.00   21.72   26.00   52.00
## Column: FreeThrowPercent
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.7198  0.7890  0.7830  0.8520  1.0000
## Column: OffensiveRebounds
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    8.00   10.00   10.55   13.00   28.00
```

```
## Column: TotalRebounds
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   25.00   39.00   43.00   43.54   48.00   74.00
## Column: Assists
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   11.00   23.00   27.00   26.67   30.00   50.00
## Column: Steals
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   6.000   7.000   7.474   9.000  20.000
## Column: Blocks
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   3.000   5.000   5.142   7.000  17.000
## Column: Turnovers
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     2.0    10.0    13.0    12.9    15.0    28.0
## Column: PersonalFouls
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    4.00   16.00   19.00   18.73   21.00   34.00
## Column: OppFieldGoals
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   26.00   38.00   42.00   42.17   46.00   65.00
## Column: OppFieldGoalAttempts
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    67.0    84.0    89.0    88.9    93.0   119.0
## Column: OppFieldGoalPercent
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2770  0.4380  0.4750  0.4752  0.5120  0.6710
## Column: OppThreePoints
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00   10.00   13.00   12.84   15.00   27.00
## Column: OppThreePointAttempts
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    12.0    30.0    35.0    35.1    39.0    63.0
## Column: OppThreePointPercent
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0690  0.3100  0.3655  0.3649  0.4170  0.6450
## Column: OppFreeThrows
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   13.00   17.00   17.03   21.00   44.00
## Column: OppFreeThrowAttempts
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   17.00   21.00   21.72   26.00   52.00
## Column: OppFreeThrowPercent
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.7198  0.7890  0.7830  0.8520  1.0000
## Column: OppOffensiveRebounds
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    8.00   10.00   10.55   13.00   28.00
## Column: OppTotalRebounds
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   25.00   39.00   43.00   43.54   48.00   74.00
## Column: OppAssists
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   11.00   23.00   27.00   26.67   30.00   50.00
```

```
## Column: OppSteals
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   6.000   7.000   7.474   9.000  20.000
## Column: OppBlocks
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   3.000   5.000   5.142   7.000  17.000
## Column: OppTurnovers
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     2.0    10.0    13.0    12.9    15.0    28.0
## Column: OppPersonalFouls
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    4.00   16.00   19.00   18.73   21.00   34.00
## Column: OpponentWins
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.000   1.000   1.575   2.000   6.000
## Column: OpponentLosses
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.000   1.000   1.575   2.000   6.000
```

```
for (col in names(NBA_Game_Log)) {
  cat("Column:", col, ", NA count:", sum(is.na(NBA_Game_Log[[col]])), "\n")
}
```

```
## Column: Team_Index , NA count: 0
## Column: Date , NA count: 0
## Column: Opp , NA count: 0
## Column: WinLoss , NA count: 0
## Column: Points , NA count: 0
## Column: OppPoints , NA count: 0
## Column: FieldGoals , NA count: 0
## Column: FieldGoalAttempts , NA count: 0
## Column: FieldGoalPercent , NA count: 0
## Column: ThreePoints , NA count: 0
## Column: ThreePointAttempts , NA count: 0
## Column: ThreePointPercent , NA count: 0
## Column: FreeThrows , NA count: 0
## Column: FreeThrowAttempts , NA count: 0
## Column: FreeThrowPercent , NA count: 0
## Column: OffensiveRebounds , NA count: 0
## Column: TotalRebounds , NA count: 0
## Column: Assists , NA count: 0
## Column: Steals , NA count: 0
## Column: Blocks , NA count: 0
## Column: Turnovers , NA count: 0
## Column: PersonalFouls , NA count: 0
## Column: OppFieldGoals , NA count: 0
## Column: OppFieldGoalAttempts , NA count: 0
## Column: OppFieldGoalPercent , NA count: 0
## Column: OppThreePoints , NA count: 0
## Column: OppThreePointAttempts , NA count: 0
## Column: OppThreePointPercent , NA count: 0
## Column: OppFreeThrows , NA count: 0
## Column: OppFreeThrowAttempts , NA count: 0
## Column: OppFreeThrowPercent , NA count: 0
```

```
## Column: OppOffensiveRebounds , NA count: 0
## Column: OppTotalRebounds , NA count: 0
## Column: OppAssists , NA count: 0
## Column: OppSteals , NA count: 0
## Column: OppBlocks , NA count: 0
## Column: OppTurnovers , NA count: 0
## Column: OppPersonalFouls , NA count: 0
## Column: Opponent , NA count: 0
## Column: OpponentWins , NA count: 0
## Column: OpponentLosses , NA count: 0
## Column: OpponentStrength , NA count: 0
## Column: Team , NA count: 0
## Column: TeamWinLoss , NA count: 0
```

```
for (col in names(NBA_Game_Log)) {
  if (is.numeric(NBA_Game_Log[[col]])) {
    non_numeric <- !grepl("^-?\\d+\\.?\\d*$", NBA_Game_Log[[col]])
    cat("Column:", col, ", Non-numeric count:", sum(non_numeric), "\n")
  }
}
```

```
## Column: Team_Index , Non-numeric count: 0
## Column: Points , Non-numeric count: 0
## Column: OppPoints , Non-numeric count: 0
## Column: FieldGoals , Non-numeric count: 0
## Column: FieldGoalAttempts , Non-numeric count: 0
## Column: FieldGoalPercent , Non-numeric count: 0
## Column: ThreePoints , Non-numeric count: 0
## Column: ThreePointAttempts , Non-numeric count: 0
## Column: ThreePointPercent , Non-numeric count: 0
## Column: FreeThrows , Non-numeric count: 0
## Column: FreeThrowAttempts , Non-numeric count: 0
## Column: FreeThrowPercent , Non-numeric count: 0
## Column: OffensiveRebounds , Non-numeric count: 0
## Column: TotalRebounds , Non-numeric count: 0
## Column: Assists , Non-numeric count: 0
## Column: Steals , Non-numeric count: 0
## Column: Blocks , Non-numeric count: 0
## Column: Turnovers , Non-numeric count: 0
## Column: PersonalFouls , Non-numeric count: 0
## Column: OppFieldGoals , Non-numeric count: 0
## Column: OppFieldGoalAttempts , Non-numeric count: 0
## Column: OppFieldGoalPercent , Non-numeric count: 0
## Column: OppThreePoints , Non-numeric count: 0
## Column: OppThreePointAttempts , Non-numeric count: 0
## Column: OppThreePointPercent , Non-numeric count: 0
## Column: OppFreeThrows , Non-numeric count: 0
## Column: OppFreeThrowAttempts , Non-numeric count: 0
## Column: OppFreeThrowPercent , Non-numeric count: 0
## Column: OppOffensiveRebounds , Non-numeric count: 0
## Column: OppTotalRebounds , Non-numeric count: 0
## Column: OppAssists , Non-numeric count: 0
## Column: OppSteals , Non-numeric count: 0
## Column: OppBlocks , Non-numeric count: 0
```

```
## Column: OppTurnovers , Non-numeric count: 0
## Column: OppPersonalFouls , Non-numeric count: 0
## Column: OpponentWins , Non-numeric count: 0
## Column: OpponentLosses , Non-numeric count: 0
```

```
for (col in names(NBA_Game_Log)) {
  na_rows <- which(is.na(NBA_Game_Log[[col]]))
  if (length(na_rows) > 0) {
    cat("Column:", col, ", NA rows:", na_rows, "\n")
  }
}

for (col in names(NBA_Game_Log)) {
  if (is.numeric(NBA_Game_Log[[col]])) {
    non_numeric_rows <- which(grepl("[^0-9.]", NBA_Game_Log[[col]]))
    if (length(non_numeric_rows) > 0) {
      cat("Column:", col, ", Non-numeric rows:", non_numeric_rows, "\n")
    }
  }
}

for (col in names(NBA_Game_Log)) {
  if (is.numeric(NBA_Game_Log[[col]])) {
    non_numeric_rows <- which(!grepl("^-?\\d+\\.?\\d*$", NBA_Game_Log[[col]]))
    if (length(non_numeric_rows) > 0) {
      cat("Column:", col, "\n")
      print(NBA_Game_Log[non_numeric_rows, ])
    } else {
      cat("No non-numeric values found in column:", col, "\n")
    }
  }
}
```

```
## No non-numeric values found in column: Team_Index
## No non-numeric values found in column: Points
## No non-numeric values found in column: OppPoints
## No non-numeric values found in column: FieldGoals
## No non-numeric values found in column: FieldGoalAttempts
## No non-numeric values found in column: FieldGoalPercent
## No non-numeric values found in column: ThreePoints
## No non-numeric values found in column: ThreePointAttempts
## No non-numeric values found in column: ThreePointPercent
## No non-numeric values found in column: FreeThrows
## No non-numeric values found in column: FreeThrowAttempts
## No non-numeric values found in column: FreeThrowPercent
## No non-numeric values found in column: OffensiveRebounds
## No non-numeric values found in column: TotalRebounds
## No non-numeric values found in column: Assists
## No non-numeric values found in column: Steals
## No non-numeric values found in column: Blocks
## No non-numeric values found in column: Turnovers
## No non-numeric values found in column: PersonalFouls
## No non-numeric values found in column: OppFieldGoals
## No non-numeric values found in column: OppFieldGoalAttempts
```

```
## No non-numeric values found in column: OppFieldGoalPercent
## No non-numeric values found in column: OppThreePoints
## No non-numeric values found in column: OppThreePointAttempts
## No non-numeric values found in column: OppThreePointPercent
## No non-numeric values found in column: OppFreeThrows
## No non-numeric values found in column: OppFreeThrowAttempts
## No non-numeric values found in column: OppFreeThrowPercent
## No non-numeric values found in column: OppOffensiveRebounds
## No non-numeric values found in column: OppTotalRebounds
## No non-numeric values found in column: OppAssists
## No non-numeric values found in column: OppSteals
## No non-numeric values found in column: OppBlocks
## No non-numeric values found in column: OppTurnovers
## No non-numeric values found in column: OppPersonalFouls
## No non-numeric values found in column: OpponentWins
## No non-numeric values found in column: OpponentLosses
```

6.5.4 The dataframe is now clean

6.6 Results

6.6.1 Our final check

6.6.2 Once again our Team_Index column should be numeric. Change that.

```
NBA_Results$Team_Index <- as.numeric(NBA_Results$Team_Index)
```

```
for (col in names(NBA_Results)) {
  cat("Column:", col, ", Class:", class(NBA_Results[[col]]), "\n")
}
```

```
## Column: Team_Index , Class: numeric
## Column: Date , Class: Date
## Column: HomeAway , Class: logical
## Column: Opponent , Class: character
## Column: Winloss , Class: logical
## Column: Overtime , Class: logical
## Column: Points , Class: numeric
## Column: OppPoints , Class: numeric
## Column: OpponentWins , Class: integer
## Column: OpponentLosses , Class: integer
## Column: Team , Class: character
## Column: TeamColor , Class: character
```

```
for (col in names(NBA_Results)) {
  if (is.numeric(NBA_Results[[col]])) {
    cat("Column:", col, "\n")
    print(summary(NBA_Results[[col]]))
  }
}
```

```
## Column: Team_Index
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0     8.0    16.0    15.5    23.0    30.0
## Column: Points
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    73.0   105.0   114.0   114.2   123.0   157.0
## Column: OppPoints
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    73.0   105.0   114.0   114.2   123.0   157.0
## Column: OpponentWins
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.000   1.000   1.579   2.000   6.000
## Column: OpponentLosses
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.000   1.000   1.579   2.000   6.000
```

```
for (col in names(NBA_Results)) {
  cat("Column:", col, ", NA count:", sum(is.na(NBA_Results[[col]])), "\n")
}
```

```
## Column: Team_Index , NA count: 0
## Column: Date , NA count: 0
## Column: HomeAway , NA count: 0
## Column: Opponent , NA count: 0
## Column: Winloss , NA count: 0
## Column: Overtime , NA count: 0
## Column: Points , NA count: 0
## Column: OppPoints , NA count: 0
## Column: OpponentWins , NA count: 0
## Column: OpponentLosses , NA count: 0
## Column: Team , NA count: 0
## Column: TeamColor , NA count: 0
```

```
for (col in names(NBA_Results)) {
  if (is.numeric(NBA_Results[[col]])) {
    non_numeric <- !grepl("^-?\\d+\\.?\\d*$", NBA_Results[[col]])
    cat("Column:", col, ", Non-numeric count:", sum(non_numeric), "\n")
  }
}
```

```
## Column: Team_Index , Non-numeric count: 0
## Column: Points , Non-numeric count: 0
## Column: OppPoints , Non-numeric count: 0
## Column: OpponentWins , Non-numeric count: 0
## Column: OpponentLosses , Non-numeric count: 0
```

```
for (col in names(NBA_Results)) {
  na_rows <- which(is.na(NBA_Results[[col]]))
  if (length(na_rows) > 0) {
    cat("Column:", col, ", NA rows:", na_rows, "\n")
  }
}
```

```r
for (col in names(NBA_Results)) {
  if (is.numeric(NBA_Results[[col]])) {
    non_numeric_rows <- which(grepl("[^0-9.]", NBA_Results[[col]]))
    if (length(non_numeric_rows) > 0) {
      cat("Column:", col, ", Non-numeric rows:", non_numeric_rows, "\n")
    }
  }
}

for (col in names(NBA_Results)) {
  if (is.numeric(NBA_Results[[col]])) {
    non_numeric_rows <- which(!grepl("^-?\\d+\\.?\\d*$", NBA_Results[[col]]))
    if (length(non_numeric_rows) > 0) {
      cat("Column:", col, "\n")
      print(NBA_Results[non_numeric_rows, ])
    } else {
      cat("No non-numeric values found in column:", col, "\n")
    }
  }
}
```

```
## No non-numeric values found in column: Team_Index
## No non-numeric values found in column: Points
## No non-numeric values found in column: OppPoints
## No non-numeric values found in column: OpponentWins
## No non-numeric values found in column: OpponentLosses
```

7 Primary & Unique Keys

7.1 Player ID

7.1.1 Generate the player IDs to use as a Primary Key for the `Roster`, `Per_Game` and `Totals` dfs.

7.1.2 Using mapping of team abbreviations to team numbers. Merge the team numbers with NBA_Roster and order sequentially to generate the Player_Id number. Convert Player_ID to numeric (if necessary).

7.1.3 Then, view NBA_Roster to ensure Player_ID is correctly generated.

```r
team_numbers <- data.frame(
  Team = c("ATL", "BOS", "BRK", "CHA", "CHI", "CLE", "DAL", "DEN", "DET", "GSW",
           "HOU", "IND", "LAC", "LAL", "MEM", "MIA", "MIL", "MIN", "NOP", "NYK",
           "OKC", "ORL", "PHI", "PHO", "POR", "SAC", "SAS", "TOR", "UTA", "WAS"),
  Team_Number = c("01", "02", "03", "04", "05", "06", "07", "08", "09", "10",
                  "11", "12", "13", "14", "15", "16", "17", "18", "19", "20",
                  "21", "22", "23", "24", "25", "26", "27", "28", "29", "30")
)
```

```
NBA_Roster <- merge(NBA_Roster, team_numbers, by = "Team", all.x = TRUE)

NBA_Roster <- NBA_Roster %>%
  group_by(Team) %>%
  mutate(Sequential_ID = row_number()) %>%
  ungroup() %>%
  mutate(Player_ID = paste0(Team_Number, sprintf("%03d", Sequential_ID)))

NBA_Roster$Player_ID <- as.numeric(gsub("[^0-9]", "", NBA_Roster$Player_ID))

head(NBA_Roster)
```

```
## # A tibble: 6 x 26
##   Team  Team_Index Country.abr Player      FirstName LastName Position Height
##   <chr> <chr>      <chr>       <chr>       <chr>     <chr>    <chr>     <dbl>
## 1 ATL   1          ao          bruno fern~ bruno     fernando c            81
## 2 ATL   1          ch          clint cape~ clint     capela   c            82
## 3 ATL   1          cz          vit krejci  vit       krejci   pg           80
## 4 ATL   1          rs          bogdan bog~ bogdan    bogdano~ sg           77
## 5 ATL   1          sn          mouhamed g~ mouhamed  gueye    pf           83
## 6 ATL   1          us          dejounte m~ dejounte  murray   sg           77
## # i 18 more variables: Weight <dbl>, BirthDate <dttm>, College <chr>,
## #   Address <chr>, City <chr>, State <chr>, Zip <dbl>, County <chr>,
## #   Latitude <dbl>, Longitude <dbl>, Country <chr>, Country.abr3 <chr>,
## #   Country.num <int>, Cty.Latitude <dbl>, Cty.Longitude <dbl>,
## #   Team_Number <chr>, Sequential_ID <int>, Player_ID <dbl>
```

```
NBA_Roster <- NBA_Roster %>% select(-c(24, 25))

colnames(NBA_Roster)[3] <- "Country_abr"
colnames(NBA_Roster)[20] <- "Country_abr3"
colnames(NBA_Roster)[21] <- "Country_num"
colnames(NBA_Roster)[22] <- "Cty_Latitude"
colnames(NBA_Roster)[23] <- "Cty_Longitude"
```

7.1.4 Now, we join the Player_Ids with the `Per_Game` and check for missing Player_IDs.

7.1.5 Then identify rows with missing Player_IDs.

```
NBA_Per_Game <- NBA_Per_Game %>%
  group_by(Player) %>%
  filter(row_number(desc(Team_Index)) == 1) %>%
  ungroup()

NBA_Per_Game <- NBA_Per_Game %>%
  left_join(NBA_Roster %>% select(Player, Player_ID), by = "Player")

missing_ids <- sum(is.na(NBA_Per_Game$Player_ID))
if (missing_ids > 0) {
  cat(paste("Warning: There are", missing_ids, "players with missing Player_IDs after merging.\n"))
}
```

```
## Warning: There are 52 players with missing Player_IDs after merging.
```

```
missing_ids_df <- NBA_Per_Game %>% filter(is.na(Player_ID))
```

8 Merge team numbers with NBA_Roster.

9 Identify players with missing Player_IDs. Then, generate Player IDs for missing players starting after 20.

10 Merge updated Player IDs back into NBA_Per_Game.

11 Check for remaining missing Player IDs

```
NBA_Per_Game <- merge(NBA_Per_Game, team_numbers, by = "Team", all.x = TRUE)

missing_players <- NBA_Per_Game %>%
  filter(is.na(Player_ID)) %>%
  distinct(Player)

missing_players <- missing_players %>%
  left_join(NBA_Per_Game %>% select(Player, Team_Number) %>% distinct(), by = "Player") %>%
  group_by(Team_Number) %>%
  mutate(
    Sequential_ID = row_number() + 20,
    Player_ID = paste0(Team_Number, sprintf("%03d", Sequential_ID))
  ) %>%
  ungroup()

NBA_Per_Game <- NBA_Per_Game %>%
  left_join(missing_players %>% select(Player, Player_ID), by = "Player")

NBA_Per_Game$Player_ID.y <- as.numeric(NBA_Per_Game$Player_ID.y)

NBA_Per_Game <- NBA_Per_Game %>%
  mutate(Player_ID = coalesce(Player_ID.x, Player_ID.y)) %>%
  select(-Player_ID.x, -Player_ID.y)

missing_ids <- sum(is.na(NBA_Per_Game$Player_ID))
if (missing_ids > 0) {
  cat(paste("Warning: There are", missing_ids, "players with missing Player_IDs after merging.\n"))
}
```

11.0.1 Lets do some final df clean up, also let's seperate the players names to have uniformaty between dfs.

```r
NBA_Per_Game <- NBA_Per_Game %>%
  separate(Player, into = c("FirstName", "LastName"), sep = " ", remove = FALSE)
```

```
## Warning: Expected 2 pieces. Additional pieces discarded in 38 rows [43, 46, 61, 91,
## 100, 103, 108, 110, 120, 160, 173, 213, 244, 254, 256, 257, 258, 275, 293,
## 296, ...].
```

```r
colnames(NBA_Per_Game)[7] <- "GamesPlayed"
colnames(NBA_Per_Game)[9] <- "MinutesPlayed"
colnames(NBA_Per_Game)[19] <- "EffectiveFieldGoalPercent"

NBA_Per_Game <- NBA_Per_Game %>% select(-c(32, 35))
```

12 Check the number of unique players to confirm no duplicates.

```r
n_distinct(NBA_Roster$Player)
```

```
## [1] 530
```

```r
n_distinct(NBA_Per_Game$Player)
```

```
## [1] 572
```

12.0.1 Do the same for the `Totals` df.

```r
NBA_Totals <- NBA_Totals %>%
  group_by(Player) %>%
  filter(row_number(desc(Team_Index)) == 1) %>%
  ungroup()

NBA_Totals <- NBA_Totals %>%
  left_join(NBA_Per_Game %>% select(Player, Player_ID), by = "Player")

missing_ids <- sum(is.na(NBA_Totals$Player_ID))
if (missing_ids > 0) {
  cat(paste("Warning: There are", missing_ids, "players with missing Player_IDs after merging.\n"))
}

NBA_Totals <- NBA_Totals %>%
  separate(Player, into = c("FirstName", "LastName"), sep = " ", remove = FALSE)
```

```
## Warning: Expected 2 pieces. Additional pieces discarded in 38 rows [38, 39, 55, 91,
## 99, 101, 103, 115, 118, 160, 169, 212, 236, 239, 240, 245, 248, 264, 292,
## 296, ...].
```

```
duplicated_ids_df <- NBA_Totals %>% filter(duplicated(Player))

n_distinct(NBA_Totals$Player)
```

```
## [1] 572
```

12.0.2 For data visualization purposes we need to create a long table based on player performace. That way we can use stats such as PointsPerGame as a category.

12.0.3 Identify top performers in points per game, rebounds, and assists.

12.0.4 Pivot the data frame.

12.0.5 Display top performers.

```
player_performance <- NBA_Per_Game %>%
  select(Team, Player_ID, Player, PointsPerGame, TotalReboundsPerGame, AssistsPerGame, StealsPerGame, Bl
  arrange(desc(PointsPerGame))

player_performance_long <- player_performance %>%
  pivot_longer(
    cols = PointsPerGame:BlocksPerGame,
    names_to = "Stats",
    values_to = "Value"
  )

print(player_performance_long)
```

```
## # A tibble: 2,860 x 6
##    Team  Player_ID Player      Team_Index Stats                Value
##    <chr>     <dbl> <chr>            <dbl> <chr>                <dbl>
##  1 PHI       23002 joel embiid         23 PointsPerGame         34.7
##  2 PHI       23002 joel embiid         23 TotalReboundsPerGame  11
##  3 PHI       23002 joel embiid         23 AssistsPerGame         5.6
##  4 PHI       23002 joel embiid         23 StealsPerGame          1.2
##  5 PHI       23002 joel embiid         23 BlocksPerGame          1.7
##  6 DAL        7008 luka dončić          7 PointsPerGame         33.9
##  7 DAL        7008 luka dončić          7 TotalReboundsPerGame   9.2
##  8 DAL        7008 luka dončić          7 AssistsPerGame         9.8
##  9 DAL        7008 luka dončić          7 StealsPerGame          1.4
## 10 DAL        7008 luka dončić          7 BlocksPerGame          0.5
## # i 2,850 more rows
```

```
player_performance_long$Team_Index <- as.numeric(player_performance_long$Team_Index)
```

12.0.6 Also, do the same for `Player_Performance_Long`

```
player_performance_long <- player_performance_long %>%
  group_by(Player) %>%
  filter(row_number(desc(Team_Index)) == 1) %>%
  ungroup()

player_performance_long <- player_performance_long %>%
  left_join(NBA_Per_Game %>% select(Player, Player_ID), by = "Player")

missing_ids <- sum(is.na(player_performance_long$Player_ID))
```

```
## Warning: Unknown or uninitialised column: `Player_ID`.
```

```
if (missing_ids > 0) {
  cat(paste("Warning: There are", missing_ids, "players with missing Player_IDs after merging.\n"))
}

player_performance_long <- player_performance_long %>%
  separate(Player, into = c("FirstName", "LastName"), sep = " ", remove = FALSE)
```

```
## Warning: Expected 2 pieces. Additional pieces discarded in 38 rows [35, 75, 92, 99,
## 101, 104, 112, 114, 116, 122, 125, 142, 159, 179, 185, 203, 208, 275, 281,
## 296, ...].
```

```
duplicated_ids_df <- player_performance_long %>% filter(duplicated(Player))

n_distinct(player_performance_long$Player)
```

```
## [1] 572
```

12.1 Game ID

12.1.1 We can bring the Player_ID over to the `Starting_Lineups` df as a unique key as well as generating a Game_ID.

12.1.2 Perform left join with many-to-many relationship.

12.1.3 Check for missing Player_IDs

```
NBA_Starting_Lineups <- NBA_Starting_Lineups %>%
  left_join(NBA_Per_Game %>% select(LastName, Player_ID), by = "LastName", relationship = "many-to-many

missing_ids <- sum(is.na(NBA_Starting_Lineups$Player_ID))
if (missing_ids > 0) {
  cat(paste("Warning: There are", missing_ids, "players with missing Player_IDs after merging.\n"))
}

duplicated_ids_df <- NBA_Starting_Lineups %>% filter(duplicated(Player_ID, Date))

NBA_Starting_Lineups <- NBA_Starting_Lineups %>%
  distinct(Player_ID, Date, .keep_all = TRUE)
```

12.1.4 Function to create Game_ID

12.1.5 Convert date to numeric (number of days since a specific origin date). Combine Team_Index and numeric date to create a unique Game_ID.

12.1.6 Apply the function to create Game_ID in NBA_Results.

```
generate_game_id <- function(team_index, date) {
  numeric_date <- as.numeric(as.Date(date) - as.Date("2000-01-01"))
  game_id <- team_index * 100000 + numeric_date
  return(game_id)
}

NBA_Results <- NBA_Results %>%
  mutate(Game_ID = generate_game_id(Team_Index, Date))
```

12.1.7 Because the game dates and team numbers are consistant we a run our ID function on the `Game_Log` and `Starting_Lineups`

```
NBA_Game_Log <- NBA_Game_Log %>%
  mutate(Game_ID = generate_game_id(Team_Index, Date))

NBA_Starting_Lineups <- NBA_Starting_Lineups %>%
  mutate(Game_ID = generate_game_id(Team_Index, Date))
```

12.1.8 Merge team names into NBA_Game_Log

```
NBA_Game_Log <- NBA_Game_Log %>%
  mutate(Opponent = team_mapping[Opp])
```

13 Oppnent Team Performance

13.1 Performance of Teams based on Tiers

13.1.1 Step 1: Calculate Opponent Win-Loss Records. Ensure to handle missing values. Remove leading/trailing whitespace in column names.

13.1.2 Step 2: Merge win-loss records with NBA_Results. Lowercase NBA_Game_Log Opponent column.

13.1.3 Step 3: Merge NBA_Game_Log with enhanced NBA_Results.

13.1.4 Step 4: Categorize Opponents into Tiers.

```
Opponent_Records <- NBA_Results %>%
  group_by(Opponent) %>%
  summarize(
    OpponentWins = sum(!Winloss, na.rm = TRUE),
    OpponentLosses = sum(Winloss, na.rm = TRUE)
  )

names(NBA_Results) <- trimws(names(NBA_Results))
names(Opponent_Records) <- trimws(names(Opponent_Records))
names(NBA_Game_Log) <- trimws(names(NBA_Game_Log))

NBA_Results <- NBA_Results %>%
  left_join(Opponent_Records, by = "Opponent")

NBA_Game_Log$Opponent <- tolower(NBA_Game_Log$Opponent)

NBA_Game_Log <- NBA_Game_Log %>%
  left_join(NBA_Results %>% select(Date, Opponent, OpponentWins.x, OpponentLosses.x), by = c("Date", "Op

NBA_Game_Log <- NBA_Game_Log %>%
  mutate(OpponentStrength = case_when(
    OpponentWins / (OpponentWins + OpponentLosses) >= 0.6 ~ "Strong",
    OpponentWins / (OpponentWins + OpponentLosses) <= 0.4 ~ "Weak",
    TRUE ~ "Average"
  ))

NBA_Game_Log <- NBA_Game_Log %>% select(-c(46, 47))

NBA_Results <- NBA_Results %>% select(-c(14, 15))
```

13.1.5 Final df cleanup

13.1.6 Display the resulting data frame

```
colnames(NBA_Results)[9] <- "OpponentWins"
colnames(NBA_Results)[10] <- "OpponentLosses"

print(NBA_Game_Log)
```

```
## # A tibble: 2,460 x 45
##    Team_Index Date       Opp   WinLoss Points OppPoints FieldGoals
##         <dbl> <date>     <chr> <lgl>    <dbl>     <dbl>      <dbl>
##  1          1 2023-10-25 cho   FALSE      110       116         39
##  2          1 2023-10-27 nyk   FALSE      120       126         42
##  3          1 2023-10-29 mil   TRUE       127       110         47
##  4          1 2023-10-30 min   TRUE       127       113         48
##  5          1 2023-11-01 was   TRUE       130       121         46
##  6          1 2023-11-04 nop   TRUE       123       105         45
##  7          1 2023-11-06 okc   FALSE      117       126         38
##  8          1 2023-11-09 orl   TRUE       120       119         41
```

```
##  9          1 2023-11-11 mia   FALSE      109       117         38
## 10          1 2023-11-14 det   TRUE       126       120         45
## # i 2,450 more rows
## # i 38 more variables: FieldGoalAttempts <dbl>, FieldGoalPercent <dbl>,
## #   ThreePoints <dbl>, ThreePointAttempts <dbl>, ThreePointPercent <dbl>,
## #   FreeThrows <dbl>, FreeThrowAttempts <dbl>, FreeThrowPercent <dbl>,
## #   OffensiveRebounds <dbl>, TotalRebounds <dbl>, Assists <dbl>,
## #   Steals <dbl>, Blocks <dbl>, Turnovers <dbl>, PersonalFouls <dbl>,
## #   OppFieldGoals <dbl>, OppFieldGoalAttempts <dbl>, ...
```

14 Archive

14.1 Create .CSV files

14.1.1 This is, to be optimal for any loading preference and to archive in our records.

14.1.2 In most Data Warehouse and Business Datalakes .csv files are optimal for performance

14.1.3 Archive File_Path = "D:\Archive\"

```
folder_path <- "D:\\Archive"
file_name <- "NBA_Roster"
file_path <- paste0(folder_path, "\\", file_name, ".csv")
NBA_Roster <- as.data.frame(NBA_Roster)
write.csv(NBA_Roster,file = file_path, row.names = FALSE)

file_name <- "NBA_Starting_Lineups"
file_path <- paste0(folder_path, "\\", file_name, ".csv")
NBA_Starting_Lineups <- as.data.frame(NBA_Starting_Lineups)
write.csv(NBA_Starting_Lineups,file = file_path, row.names = FALSE)

file_name <- "NBA_Per_Game"
file_path <- paste0(folder_path, "\\", file_name, ".csv")
NBA_Per_Game <- as.data.frame(NBA_Per_Game)
write.csv(NBA_Per_Game,file = file_path, row.names = FALSE)

file_name <- "NBA_Totals"
file_path <- paste0(folder_path, "\\", file_name, ".csv")
NBA_Totals <- as.data.frame(NBA_Totals)
write.csv(NBA_Totals,file = file_path, row.names = FALSE)

file_name <- "NBA_Game_Log"
file_path <- paste0(folder_path, "\\", file_name, ".csv")
NBA_Game_Log <- as.data.frame(NBA_Game_Log)
write.csv(NBA_Game_Log,file = file_path, row.names = FALSE)

file_name <- "NBA_Results"
file_path <- paste0(folder_path, "\\", file_name, ".csv")
NBA_Results <- as.data.frame(NBA_Results)
write.csv(NBA_Results,file = file_path, row.names = FALSE)
```

15 Conclusion

15.1 ETL - Transform

15.1.1 In this overview of the Transform portion of my NBA Data pipeline we have discussed how to Load Required Packages, Load Functions, Clean the Data, Combine the Data, Run a Final Multi-layered Check and how to Archive the Data for storage.

15.2 Portforlio Project

15.2.1 Links to the entire Basketball Data Project ** including the Python Web Scrape Code and Tableau Dashboards can be found on my GitHub Profile