

ETL



Data Warehousing (ETL) TRANSFORM

Sawandi Kirby

2024-05-04

Contents

| | |
|--|----------|
| 1 ETL -(Extract Transform Load) | 3 |
| 1.1 Definition | 3 |
| 2 Extract | 3 |
| 2.1 Python- Web Scrape | 3 |
| 3 Import data | 4 |
| 3.1 Load Required Packages | 4 |
| 3.2 Load Functions | 4 |
| 3.3 Import Datasets | 5 |
| 4 Clean Data | 5 |
| 4.1 Full list | 6 |
| 4.2 Roster_df | 6 |
| 4.3 Per_Game_df | 7 |
| 4.4 Totals_df | 8 |
| 4.5 Game_Log_df | 9 |

| | | |
|-----------|---|-----------|
| 4.6 | Results_df | 10 |
| 4.7 | Starting_lineup..... | 10 |
| 5 | Combine Data | 12 |
| 5.1 | Bind Rows | 12 |
| 5.2 | Assign Positions | 13 |
| 5.3 | Build a function that we can run our dfs through..... | 15 |
| 5.4 | Per_Game | 15 |
| 5.5 | Totals | 15 |
| 6 | Archive | 16 |
| 6.1 | Create .csv | 16 |
| 7 | Remove Objects | 17 |
| 7.1 | Use the rm function..... | 17 |
| 8 | Combine Data Continued... | 19 |
| 8.1 | Roster | 19 |
| 9 | Final Check | 20 |
| 9.1 | Per_Game Check | 20 |
| 9.2 | Multi-layered Check | 23 |
| 9.3 | Results Check | 29 |
| 10 | Archive Continued... | 32 |
| 10.1 | Create .CSV files | 32 |
| 10.2 | Create .XLSX files | 32 |
| 11 | Conclusion | 35 |
| 11.1 | ETL | 35 |
| 11.2 | Portforlio Project | 35 |

1 ETL -(Extract Transform Load)

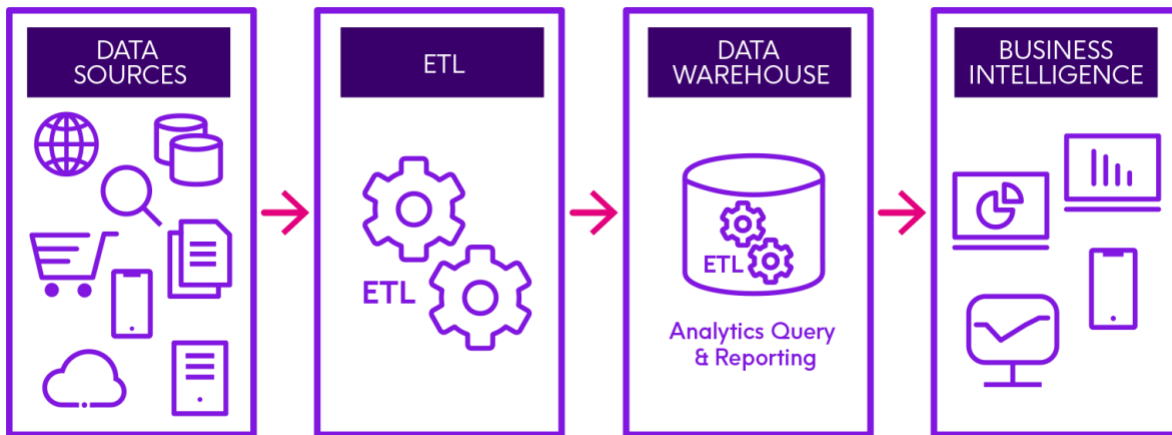


Figure 1: Extract Transform Load

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

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

1.1.4 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.

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

1.1.6 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)
library(readxl)
library(openxlsx)
library(dplyr)
library(stringr)
library(htmltools)
library(httputil)
library(googleCloudStorage)
library(googleAuthR)
library(bigrquery)
```

3.2 Load Functions

3.2.1 Function to read Excel file sheets.

```
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)
}
```

3.2.2 Function to trim rows of character columns in a data frame.

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

3.2.3 Function to convert height to inches.

```
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)
}
```

3.2.4 Function to convert height to inches.

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

* 12 .
as.inte

3.3 Import Datasets

3.3.1 Use read_excel_file to extract Sheets from Excel file, into a list with the appropriate Team abbreviation.

```
ATL <-
read_excel_file("C:\\Users\\kirby\\OneDrive\\Desktop\\NBA\\NBA_2022\\ATL_data.xlsx")
```

3.3.2 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]]
```

4 Clean Data

4.1 Full list

4.1.1 First thing we are going to do is, replace all empty strings in every data frame with NA.

4.1.2 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, .))

```

4.1.3 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)

```

4.2 Roster_df

4.2.1 First start with the roster df, and get a glimpse of the dataframe and check out its structure.

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

```
head(ATL_Roster)
```

```

## # A tibble: 6 x 7
##   Player          Position Height Weight BirthDate      BirthCountry College
##   <chr>           <chr>    <chr>  <dbl> <dtm>         <chr>         <chr>
## 1 Cat Barber     pg        6-2    190 1994-07-25 00:00:00 us      nc state
## 2 Bogdan Bogdanović sg        6-5    220 1992-08-18 00:00:00 rs      <NA>
## 3 Chaundee Brown Jr. sg        6-5    215 1998-12-04 00:00:00 us      wake fore~
## 4 Clint Capela   c         6-10   240 1994-05-18 00:00:00 ch      <NA>
## 5 John Collins   pf        6-9    235 1997-09-23 00:00:00 us      wake fore~ ## 6
Sharife Cooper    pg        6-1    180 2001-06-11 00:00:00 us      auburn

```

4.2.3 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.

4.2.4 First remove (TW) tag from Player names. Then use the separate function.

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

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

```
## Warning: Expected 2 pieces. Additional pieces discarded in 1 rows [3].
```

4.2.5 Now we can use the trimws function to first remove the whitespace around any string columns, in our df.

4.2.6 Then use the tolower function to transform our string values into all lowercase 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))
```

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

4.2.8 The Height column is in character format although it would be easier to calculate if they were in numeric format. So let's use our function to convert the height column.

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

4.3 Per_Game_df

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

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

```
head(ATL_Per_Game)
```

```
## # A tibble: 6 x 27
##   PlayerAge GamesPlayedPerGame GamesStarted MinutesPlayedPerGame FieldGoalsPerGame
##   <chr>      <dbl> <chr>          <dbl>          <dbl>          <dbl>
## 1 traee      23 76          76            34.9           9.4
yo~
```

```
## 2 john 24 54 53 30.8 6.3
co~
## 3 24 53 52 29.8 4.8
de'andr~
## 4 kevin 23 74 60 29.6 4.7
h~
## 5 bogdan 29 63 27 29.3 5.4
~
## 6 23 3 2 27.7 3
chaunde~
## # i 21 more variables: FieldGoalAttemptsPerGame <dbl>,
FieldGoalPercentPerGame <dbl>,
## #ThreePointFieldGoalsPerGame <dbl>, ThreePointFieldGoalAttemptsPerGame <dbl>,
## #ThreePointFieldGoalPercentPerGame <dbl>, TwoPointFieldGoalsPerGame <dbl>,
## #TwoPointFieldGoalAttemptsPerGame <dbl>, TwoPointFieldGoalPercentPerGame <dbl>,
## # EffectiveFieldGoalPercentPerGame <dbl>, FreeThrowsPerGame
<dbl>, ## # FreeThrowAttemptsPerGame <dbl>,
FreeThrowPercentPerGame <dbl>,
## #OffensiveReboundsPerGame <dbl>, DefensiveReboundsPerGame <dbl>, ...
```

4.3.3 It seems that the only incorrect column in this data frame is the GamesPlayed column.

4.3.4 So let's go ahead and fix that...

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

4.4 Totals_df

```
head(ATL_Totals)
```

```
## # A tibble: 6 x 27
## Player Age TotalGamesPlayed TotalGamesStarted TotalMinutesPlayed
## <chr> <dbl> <dbl> <dbl> <dbl>
## 1 trae young 23 76 76 2652
## 2 kevin huerter 23 74 60 2188
## 3 clint capela 27 74 73 2042
## 4 bogdan 29 63 27 1848
bogdanović
## 5 danilo gallinari 33 66 18 1672
## 6 john collins 24 54 53 1663
## # i 22 more variables: TotalFieldGoalsPerGame <dbl>,
TotalFieldGoalAttempts <dbl>,
## #TotalFieldGoalPercent <dbl>, TotalThreePointFieldGoalsPerGame <dbl>,
## TotalThreePointFieldGoalAttempts <dbl>,
## TotalThreePointFieldGoalPercent <dbl>,
## TotalTwoPointFieldGoalsPerGame <dbl>, TotalTwoPointFieldGoalAttempts
## <dbl>,
## TotalTwoPointFieldGoalPercent <dbl>, TotalEffectiveFieldGoalPercent
## <dbl>,
```



```
## TotalFreeThrows <dbl>, TotalFreeThrowAttempts <dbl>,
# TotalFreeThrowPercent <dbl>,
## TotalOffensiveRebounds <dbl>, TotalDefensiveRebounds <dbl>, ...
#
```

4.4.1

It seems that almost all of our data frame is in correct format although there are a couple NA values scattered.

4.4.2 There is a final row that list totals for the team where applicable, but we can calculate these values on our own.

4.4.3 Let's remove it...

```
ATL_Totals <- ATL_Totals[-c(25), ]
```

4.5 Game_Log_df

```
head(ATL_Game_Log)
```

```
## # A tibble: 6 x 37
## Date          Opp WinLoss Points OppPoints FieldGoals FieldGoalAttempts
## <dtm>          <chr> <chr>   <dbl>   <dbl>       <dbl>          <dbl>
##
## 1 NA           <NA> <NA>     NA      NA         NA            NA
## 2 2021-10-21 00:00:00 dal      W      113     87      45          94
## 3 2021-10-23 00:00:00 cle      L      95     101     38          99
## 4 2021-10-25 00:00:00 det      W     122     104     46          90
## 5 2021-10-27 00:00:00 nop      W     102     99      40          96
## 6 2021-10-28 00:00:00 was      L     111     122     48          88
## # i 30 more variables: FieldGoalPercent <dbl>, ThreePoints
## # ThreePointAttempts <dbl>, ThreePointPercent <dbl>, FreeThrows
## # FreeThrowAttempts <dbl>, FreeThrowPercent <dbl>, OffensiveRebounds <dbl>,
## # TotalRebounds <dbl>, Assists <dbl>, Steals <dbl>, Blocks <dbl>, Turnovers <dbl>,
## # PersonalFouls <dbl>, OppFieldGoals <dbl>, OppFieldGoalAttempts <dbl>,
## # OppFieldGoalPercent <dbl>, OppThreePoints <dbl>,
## # OppThreePointAttempts <dbl>, OppThreePointPercent <dbl>,
## # OppFreeThrows <dbl>, OppFreeThrowAttempts <dbl>, ...
```

4.5.1 As we can see, all of the data types seem to be correct.

4.5.2 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")
```

4.5.3 Also 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)
```

4.6 Results_df

```
head(ATL_Results)
```

```
## # A tibble: 6 x 7
##   Date           HomeAway Opponent           Winloss Overtime Points OppPoints
##   <dtm>          <chr>    <chr>           <chr>   <chr>   <dbl>   <dbl>
## 1 2021-10-21 00:00:00      dallas mavericks  W      <NA>    113     87
## 2 2021-10-23 00:00:00 @    cleveland cavaliers L      <NA>    95     101
## 3 2021-10-25 00:00:00      detroit pistons   W      <NA>    122     104
## 4 2021-10-27 00:00:00 @    new orleans pelicans W      <NA>    102     99
## 5 2021-10-28 00:00:00 @    washington wizards L      <NA>    111     122
## 6 2021-10-30 00:00:00 @    philadelphia 76ers L      <NA>    94     122
```

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

4.6.2 If you look closely you can see that there are entire rows of NA values. We can go ahead and take those away.

```
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), ]
```

4.7 Starting_lineup.

4.7.1 We can continue to use head and str.

```
head(ATL_Starting_Lineup)
```

```
## # A tibble: 6 x 13
##   Date       `Start(ET)` `BoxScore` HomeAway Opponent WinLoss Overtime TeamPoints
##   <chr>      <lgl>      <lgl> <chr>    <chr>    <chr>    <chr>    <chr>
## 1 2021-10-21 NA        NA    Box Score <NA> Dallas M~ W    <NA>    113
## 2 2021-10-23 NA        NA    Box Score @    Clevealan~ L    <NA>    95
## 3 2021-10-25 NA        NA    Box Score <NA> Detroit ~ W    <NA>    122
## 4 2021-10-27 NA        NA    Box Score @    New Orle~ W    <NA>    102
## 5 2021-10-28 NA        NA    Box Score @    Washingt~ L    <NA>    111
## 6 2021-10-30 NA        NA    Box Score @    Philadel~ L    <NA>    94
## # i 4 more variables: OpponentPoints <chr>, Wins <chr>,
## # Losses <chr>, ## # StartingLineup <chr>
```

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

4.7.3 First, we need to fill in the NA values in the HomeAway and Overtime columns. Then, because they are one of two values. 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$Overtime <- ifelse(is.na(ATL_Starting_Lineup$Overtime),
"rt")

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)
```

```
ATL_Starting_Lineup
, ATL_Starting_Lineup$O
)
)
```

4.7.4 Next, we see there are multiple columns entirely with NA values, Start(ET), __, and BoxScore.

4.7.5 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` )
```

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

4.7.7 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")
```

4.7.8 Also let's not forget to remove the empty rows.

```
ATL_Starting_Lineup <- ATL_Starting_Lineup[-c(21, 42, 63), ]
```

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

5 Combine Data

5.1 Bind Rows

5.1.1 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, CHI_Roster, CHO_Roster,
DAL_Roster, DEN_Rost
```

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

5.1.3 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.

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

```
team_data <- tibble(
  Team = c("ATL", "BOS", "BRK", "CHO", "CHI", "CLE", "DAL", "DEN", "DET",
           "GSW", "HOU",
           "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, CHI_Roster, CHO_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)])

NBA_Roster$Date <- as.Date(paste("2020"), format = "%Y")
```

```
head(NBA_Roster)
```

```
## # A tibble: 6 x 12
## Team_Index Player   FirstName LastName Position Height Weight BirthDate
##   <chr>      <chr>      <chr>      <chr>      <chr>      <dbl> <dbl> <dtm>
## 1 1          cat barber cat        barber    pg         74      190 1994-07-25
##                00:00:00
## 2 1          bogdan bogd~ bogdanbogdano~ sg         77      220 1992-08-18
##                00:00:00
## 3 1          chaundee br~ chaundee brown sg         77      215 1998-12-04
##                00:00:00
## 4 1          clint capela clint  capela    c          82      240 1994-05-18
##                00:00:00
## 5 1          john collins john   collins   pf         81      235 1997-09-23
##                00:00:00
## 6 1          sharife coo~ sharifecooper pg         73      180 2001-06-11
##                00:00:00
## # i 4 more variables: BirthCountry <chr>, College <chr>, Team <chr>, Date
## <date>
```

5.1.5 Continue on to repeat the steps for *Per_Game*, *Totals*, *Starting_Lineups*, *Game_Log* and *Results...*

5.2 Assign Positions

5.2.1 Check which df needs positions to be assigned to them.

5.2.2 So far we have three data frames (*Roster*, *Per_Game*, *Totals*) that have a list of players

but only one (Roster) lists their position which can be a great filtering factor.

```
head(NBA_Per_Game)
```

```
## # A tibble: 6 x 32
##   Team_Index Player      Age GamesPlayedPerGame GamesStarted
##   <chr>      <chr>      <dbl> <chr>              <dbl>      <dbl>
## 1 1      trae young      23 76              76          34.9
## 2 1      john collins    24 54              53          30.8
## 3 1      de'andre hunter  24 53              52          29.8
## 4 1      kevin huerter     23 74              60          29.6
## 5 1      bogdan bogdano~    29 63              27          29.3
## 6 1      chaundee brown~   23 3              2          27.7
## # i 26 more variables: FieldGoalsPerGame <dbl>, FieldGoalAttemptsPerGame
## # <dbl>,
## #FieldGoalPercentPerGame <dbl>, ThreePointFieldGoalsPerGame <dbl>,
## #ThreePointFieldGoalAttemptsPerGame <dbl>, ThreePointFieldGoalPercentPerGame <dbl>,
## #TwoPointFieldGoalsPerGame <dbl>, TwoPointFieldGoalAttemptsPerGame <dbl>,
## #TwoPointFieldGoalPercentPerGame <dbl>, EffectiveFieldGoalPercentPerGame <dbl>,
## #FreeThrowsPerGame <dbl>, FreeThrowAttemptsPerGame <dbl>,
## #FreeThrowPercentPerGame <dbl>, OffensiveReboundsPerGame <dbl>, ...
```

```
head(NBA_Totals)
```

```
## # A tibble: 6 x 31
##   Team_Index Player      Age TotalGamesPlayed TotalGamesStarted TotalMinutesPlayed
##   <chr>      <chr>      <dbl>      <dbl>              <dbl>      <dbl>
## 1 1      trae young      23          76              76          26
## 2 1      kevin          23          74              60          52
## 3 1      huerter          23          74              60          21
## 4 1      clint          27          74              73          88
## 5 1      capela          29          63              27          20
## 6 1      bogdan          29          63              27          42
## 7 1      bogdan~          33          66              18          18
## 8 1      danilo          33          66              18          48
## 9 1      gallin~          24          54              53          16
## 10 1      john          24          54              53          16
## 11 1      collins          24          54              53          63
## # i 25 more variables: TotalFieldGoalsPerGame <dbl>, TotalFieldGoalAttempts
## # <dbl>,
## #TotalFieldGoalPercent <dbl>, TotalThreePointFieldGoalsPerGame <dbl>,
## #TotalThreePointFieldGoalAttempts <dbl>, TotalThreePointFieldGoalPercent <dbl>,
## #TotalTwoPointFieldGoalsPerGame <dbl>, TotalTwoPointFieldGoalAttempts <dbl>,
## #TotalTwoPointFieldGoalPercent <dbl>, TotalEffectiveFieldGoalPercent
## # <dbl>, ## # TotalFreeThrows <dbl>, TotalFreeThrowAttempts <dbl>,
## #TotalFreeThrowPercent <dbl>, ## # TotalOffensiveRebounds <dbl>,
## #TotalDefensiveRebounds <dbl>, ...
```

5.3 Build a function that we can run our dfs through.

5.3.1 To locate and assign the desired position per player.

5.3.2 Assign positions from roster dataframe to another dataframe

5.3.3 Initialize an empty vector to store positions. Loop through each player in the per_game_df. Check if the player exists in the NBA_Roster dataframe.

5.3.4 Match the player and extract the position. If the player is not found, assign NA to position. Add the positions column to the per_game_df

5.3.5 Remember to change per_game_df to totals_df for NBA_Totals.

5.4 Per_Game

```
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)  
  }
```

5.5 Totals

```
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)  
}
```

5.5.1 Call the function to assign positions to NBA_Per_Game and NBA_Totals.

```
NBA_Per_Game <- assign_positions(NBA_Per_Game)
```

```
NBA_Totals <- assign_positions_totals(NBA_Totals)
```

5.5.2 Remember to do this, for every Year...

6 Archive

6.1 Create .csv

6.1.1 Let's create .csv files because they are the most common.

6.1.2 Now that our data frames are cleaned, I like to store and archive all files so we are going to give them their own file_names to be able to easily tell them apart.

```
folder_path <- "D:\\BigQuery_example" file_name
<- "NBA_Roster_2022" 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_2022"
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_2022" 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_2022"
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_2022"
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_2022"
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)
```

6.1.3 Repeat steps for corresponding years, 2022, 2021, 2020, and so on...

7 Remove Objects

7.1 Use the rm function.

7.1.1 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
```

7.1.2 If you have been saving and archiving the files then you might have to upload them back into the system.

7.1.3 Since these files are only one sheet it is ok to use the base read_csv function.

```
NBA_Roster_2017 <- read_csv("D:\\BigQuery\\NBA_Roster_2017.csv")
```

```
## Rows: 542 Columns: 12
## -- Column specification -----
-----## Delimiter: ","
## chr (7): Player, FirstName, LastName, Position, BirthCountry, College, Team
## dbl (3): Team_Index, Height, Weight
## date (2): BirthDate, Date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

```
NBA_Roster_2018 <- read_csv("D:\\BigQuery\\NBA_Roster_2018.csv")
```

```
## Rows: 606 Columns: 12
## -- Column specification -----
-----## Delimiter: ","
## chr (7): Player, FirstName, LastName, Position, BirthCountry, College, Team
## dbl (3): Team_Index, Height, Weight
## date (2): BirthDate, Date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

```
NBA_Roster_2019 <- read_csv("D:\\BigQuery\\NBA_Roster_2019.csv")
```

```
## Rows: 622 Columns: 12
## -- Column specification -----
-----## Delimiter: ","
## chr (7): Player, FirstName, LastName, Position, BirthCountry, College, Team
```

```
## dbl (3): Team_Index, Height, Weight
## date (2): BirthDate, Date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

```
NBA_Roster_2020 <- read_csv("D:\\BigQuery\\NBA_Roster_2020.csv")
```

```
## Rows: 592 Columns: 12
## -- Column specification -----
-----## Delimiter: ","
## chr (7): Player, FirstName, LastName, Position, BirthCountry, College, Team
## dbl (3): Team_Index, Height, Weight
## date (2): BirthDate, Date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

```
NBA_Roster_2021 <- read_csv("D:\\BigQuery\\NBA_Roster_2021.csv")
```

```
## Rows: 626 Columns: 12
## -- Column specification -----
-----## Delimiter: ","
## chr (7): Player, FirstName, LastName, Position, BirthCountry, College, Team
## dbl (3): Team_Index, Height, Weight
## date (2): BirthDate, Date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

```
NBA_Roster_2022 <- read_csv("D:\\BigQuery\\NBA_Roster_2022.csv")
```

```
## Rows: 716 Columns: 12
## -- Column specification -----
-----## Delimiter: ","
## chr (7): Player, FirstName, LastName, Position, BirthCountry, College, Team
## dbl (3): Team_Index, Height, Weight
## date (2): BirthDate, Date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

```
NBA_Roster_2023 <- read_csv("D:\\BigQuery\\NBA_Roster_2023.csv")
```

```
## Rows: 532 Columns: 12
```

```
## -- Column specification -----
-----## Delimiter: ","
## chr (7): Player, FirstName, LastName, Position, BirthCountry, College, Team
## dbl (3): Team_Index, Height, Weight
## date (2): BirthDate, Date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

7.1.4 Continue this process for NBA_Per_Game, NBA_Totals, NBA_Starting_Lineups, NBA_Game_Log and NBA_Results...

8 Combine Data Continued...

8.1 Roster

8.1.1 Combine data frames for each year into one compiled data frame.

```
NBA_Roster <- bind_rows(NBA_Roster_2017, NBA_Roster_2018, NBA_Roster_2019,
NBA_Roster_2020, NBA_Roster_ head(NBA_Roster)
## # A tibble: 6 x 12
## Team_Index PlayerFirstName LastName Position Height Weight BirthDate BirthCountry

##      <dbl> <chr>   <chr>   <chr>   <chr>   <dbl> <dbl> <date> <chr>
##      1 kent ba~ kent   bazemore sf      76   195 1989-07-01 us
1
##      1 deandre~ deandre' bembry sf      77   210 1994-07-04 us
2
##      1 josé ca~ josé   calderón pg      75   200 1981-09-28 es
3
##      1 malcolm~ malcolm delaney pg      75   190 1989-03-11 us
4
##      1 mike du~ mike   dunleavy sf      81   230 1980-09-15 us
5
##      1 tim har~ tim    hardaway sg      77   205 1992-03-16 us
6
## # i 3 more variables: College <chr>, Team <chr>, Date <date>
```

8.1.2 Continue binding rows and create the rest of the data frames NBA_Per_Game, NBA_Totals, NBA_Starting_Lineups, NBA_Game_Log and NBA_Results...

9 Final Check

9.1 Per_Game Check

9.1.1 Build a function to run our data frames through. To check for NA/NULL values, nonnumeric 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.

9.1.2 In this example we are going to use NBA_Per_Game.

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

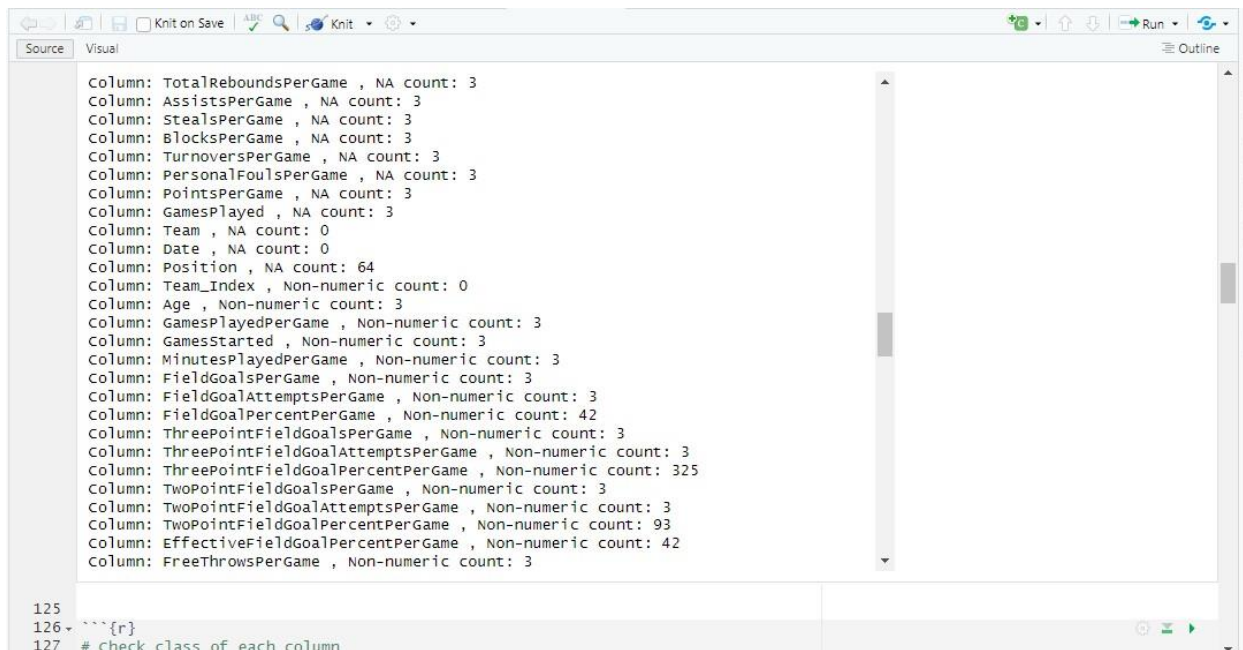


Figure 2: Per_Game Check before cleaning

9.1.4 First thing we notice is that rows 9, 15 and 28 are completely empty. Then we see that the Player column has three empty rows. 3882, 4013 and 4306.

9.1.5 They have no information on them except what team they came from.

9.1.6 We can take these entire rows out...

```
NBA_Per_Game <- NBA_Per_Game[-c(9, 15, 28, 3882, 4013, 4306), ]
```

9.1.7 In the Position column there are 61 NA values that is not many in comparison to our total data frame of 4,236 rows. So we are going to produce a list of player names that have no position value and see if we can correct them.

9.1.8 First find rows with NA values in the Position column using the subset function. Then extract names from the Player column into a new df players_with_na_positions

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

9.1.10 Make a function to fix the Positions column with NA values using a list we made from finding 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)
}
```

9.1.11 Compile the list of Players and their positions.

```

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", "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.",
                "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", "hamidou diallo", "sg")

```

"sf",

9.1.12 Execute the function to assign players their positions.

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

9.1.13 We see that almost every numeric row has NA values in them, because these are sports statistics any NA value most likely means 0, so we are going to replace the numeric values with 0.

```

NBA_Per_Game$FieldGoalPercentPerGame[is.na(NBA_Per_Game$FieldGoalPercentPerGame)] <- 0
NBA_Per_Game$ThreePointFieldGoalPercentPerGame[is.na(NBA_Per_Game$
NBA_Per_Game$TwoPointFieldGoalPercentPerGame[is.na(NBA_Per_Game$
NBA_Per_Game$EffectiveFieldGoalPercentPerGame[is.na(NBA_Per_Game$
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)
]
NBA_Per_Game$ThreePointFieldGoalAttemptsPerGame[is.na(NBA_Per_Game$
NBA_Per_Game$TwoPointFieldGoalsPerGame[is.na(NBA_Per_Game$TwoPointFieldGoalsPerGame)] <-
0
NBA_Per_Game$TwoPointFieldGoalAttemptsPerGame[is.na(NBA_Per_Game$
NBA_Per_Game$FreeThrowsPerGame[is.na(NBA_Per_Game$FreeThrowsPerGame)] <- 0
ThreePointFieldGoalPercentPerGame)] <- 0
TwoPointFieldGoalPercentPerGame)] <- 0
EffectiveFieldGoalPercentPerGame)] <- 0

ThreePointFieldGoalAttemptsPerGame)] <- 0
TwoPointFieldGoalAttemptsPerGame)] <- 0
FreeThrowAttemptsPerGame[is.na(NBA_Per_Game$FreeThrowAttemptsPerGame)] <- 0
NBA_Per_Game$OffensiveReboundsPerGame[is.na(NBA_Per_Game$OffensiveReboundsPerGame)
e)] <- 0
NBA_Per_Game$DefensiveReboundsPerGame[is.na(NBA_Per_Game$DefensiveReboundsPerGame)
e)] <- 0
NBA_Per_Game$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

```

9.2 Multi-layered Check

9.2.1 First, check the class of each column and the summary statistics for each numeric column.

9.2.2 Then, check for any NA values in each column and check for non-numeric values in each column. Then, identify the rows with NA values in each column and identify rows with non-numeric values in each numeric column.

9.2.3 Finally, display the rows with non-numeric values in each numeric column.

9.2.4 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: Team , Class: character  
## Column: Date , Class: Date  
## Column: Position , Class: character
```



```

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   8.00  15.00  15.49  23.00  30.00
## Column: Age
##   Min. 1st Qu. Median   Mean 3rd Qu.   Max.
##  19.00  23.00  25.00  25.99  29.00  43.00
## Column: GamesPlayedPerGame
##   Min. 1st Qu. Median   Mean 3rd Qu.   Max.
##   1.00  16.00  41.00  40.43  64.00  82.00
## Column: GamesStarted
##   Min. 1st Qu. Median   Mean 3rd Qu.   Max.
##   0.00   0.00   5.00  19.03  32.00  82.00
## Column: MinutesPlayedPerGame
##   Min. 1st Qu. Median   Mean 3rd Qu.   Max.
##   0.5   11.3   18.7   19.0   27.0   43.5
## Column: FieldGoalsPerGame
##   Min. 1st Qu. Median   Mean 3rd Qu.   Max.
##   0.000  1.300  2.500  3.059  4.300 11.500
## Column: FieldGoalAttemptsPerGame
##   Min. 1st Qu. Median   Mean 3rd Qu.   Max.
##   0.00   3.30   5.50   6.74   9.30  24.50
## Column: FieldGoalPercentPerGame
##   Min. 1st Qu. Median   Mean 3rd Qu.   Max.
## 0.0000 0.3950 0.4410 0.4373 0.4940 1.0000
## Column: ThreePointFieldGoalsPerGame
##   Min. 1st Qu. Median   Mean 3rd Qu.   Max.
## 0.0000 0.2000 0.7000 0.8806 1.4000 5.3000
## Column: ThreePointFieldGoalAttemptsPerGame
## Min. 1st Qu. Median Mean 3rd Qu. Max. ##
## 0.000 0.800 2.100 2.521 3.800 13.200
## Column: ThreePointFieldGoalPercentPerGame
##   Min. 1st Qu. Median   Mean 3rd Qu.   Max.
## 0.0000 0.2310 0.3330 0.2872 0.3770 1.0000
## Column: TwoPointFieldGoalsPerGame
##   Min. 1st Qu. Median   Mean 3rd Qu.   Max.
##   0.00   0.80   1.70   2.18   3.00  11.00 ## Column:
TwoPointFieldGoalAttemptsPerGame
##   Min. 1st Qu. Median   Mean 3rd Qu.   Max.
##   0.00   1.70   3.30   4.22   5.80  19.20
## Column: TwoPointFieldGoalPercentPerGame
##   Min. 1st Qu. Median   Mean 3rd Qu.   Max.
## 0.0000 0.4500 0.5040 0.4923 0.5650 1.0000
## Column: EffectiveFieldGoalPercentPerGame

```

```
##   Min. 1st Qu. Median   Mean 3rd Qu.   Max.
## 0.0000 0.4690 0.5150 0.4987 0.5600 1.5000
## Column: FreeThrowsPerGame
##   Min. 1st Qu. Median Mean 3rd Qu. Max. ##
## 0.000 0.400 0.900 1.293 1.700 10.200
## Column: FreeThrowAttemptsPerGame
##   Min. 1st Qu. Median Mean 3rd Qu. Max. ##
## 0.000 0.600 1.200 1.688 2.200 11.800
## Column: FreeThrowPercentPerGame
##   Min. 1st Qu. Median   Mean 3rd Qu.   Max.
## 0.0000 0.6415 0.7560 0.6880 0.8330 1.0000
## Column: OffensiveReboundsPerGame
##   Min. 1st Qu. Median   Mean 3rd Qu.   Max.
## 0.0000 0.3000 0.6000 0.8131 1.1000 5.4000
## Column: DefensiveReboundsPerGame
##   Min. 1st Qu. Median Mean 3rd Qu. Max. ##
## 0.000 1.300 2.300 2.641 3.500 11.400
## Column: TotalReboundsPerGame
##   Min. 1st Qu. Median   Mean 3rd Qu. Max.
## 0.000 1.700 3.000 3.451 4.600
## Column: AssistsPerGame
##   Min. 1st Qu. Median   Mean 3rd Qu. Max.
## 0.000 0.600 1.300 1.861 2.400
## Column: StealsPerGame
##   Min. 1st Qu. Median   Mean 3rd Qu. Max.
## 0.0000 0.3000 0.5000 0.6047 0.9000 2.5000
## Column: BlocksPerGame
##   Min. 1st Qu. Median   Mean 3rd Qu.   Max.
## 0.0000 0.1000 0.3000 0.3829 0.5000 6.0000
## Column: TurnoversPerGame
##   Min. 1st Qu. Median   Mean 3rd Qu.   Max.
## 0.000 0.500 0.900 1.043 1.400 5.700
## Column: PersonalFoulsPerGame
##   Min. 1st Qu. Median   Mean 3rd Qu.   Max.
## 0.000 1.000 1.600 1.624 2.200 5.000
## Column: PointsPerGame
##   Min. 1st Qu. Median Mean 3rd Qu. Max. ##
## 0.000 3.700 6.800 8.288 11.500 36.100
## Column: GamesPlayed
##   Min. 1st Qu. Median Mean 3rd Qu. Max. ##
## 1.00 16.00 41.00 40.43 64.00 82.00
```

```
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: Team , NA count: 0
## Column: Date , NA count: 0
## Column: Position , 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
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")
  }
}
}
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
```

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

9.3 Results Check

9.3.1 Let's do the same and drop a .png to show the some of the errors saving space.

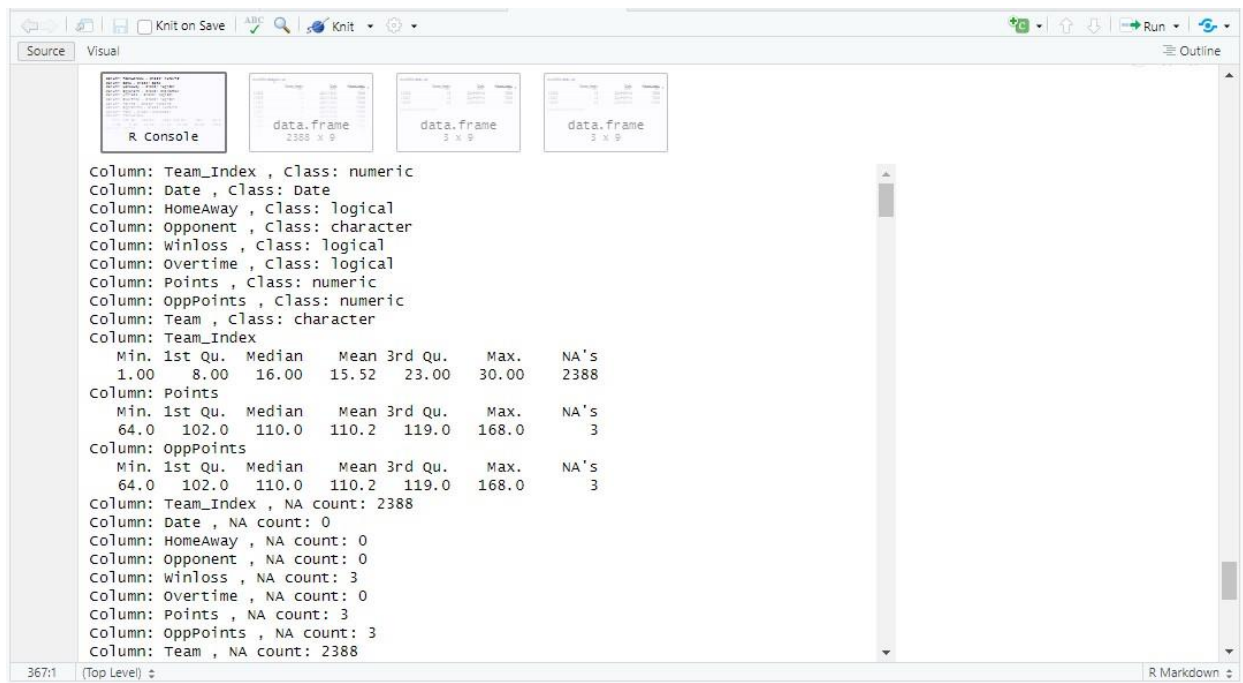


Figure 3: Results Check before cleaning

9.3.2 Fix the results of these games after looking them up.

9.3.3 First, input the row and column index. Then, the new value to assign. Finally, change the value at the specified row and column.

```

row_index <- 15463
col_index <- 5
new_value <- TRUE

NBA_Results[row_index, col_index] <- new_value

row_index <- 15463
col_index <- 7
new_value <- 112

NBA_Results[row_index, col_index] <- new_value

row_index <- 15463
col_index <- 8
new_value <- 94

NBA_Results[row_index, col_index] <- new_value

```

9.3.4 *Continue this process for rows 15711, and 16288...*

9.3.5 **Re-Run the Check.**

```

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: Team , 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      Ma
##                Qu.      x.
##   1.00   8.00  16.00  15.52  23.00  30.
## Column: Points
##   Min. 1st Qu. Median   Mean 3rd      Ma
##                Qu.      x.
##   64.0  102.0  110.0  110.2  119.0  168
## Column: OppPoints
##                .0

```

```
##   Min. 1st Qu. Median   Mean 3rd      Ma
      Qu.      x.
##   64.0  102.0  110.0  110.2  119.0  168
                                   .0
```

```
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: Team , 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
```

```
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") }
}
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
```

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

10 Archive Continued...

10.1 Create .CSV files

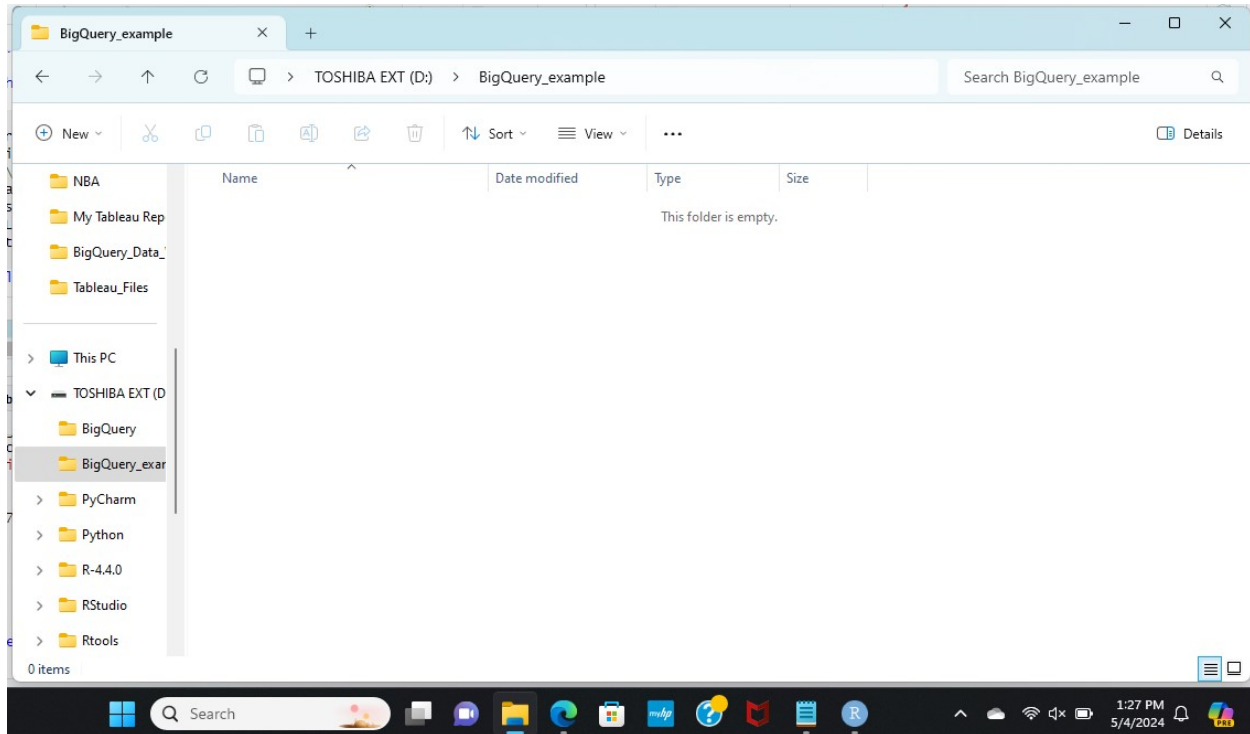


Figure 4: .CSV files before upload

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

10.1.2 In most Data Warehouse and Business Datalakes .csv files are optimal for performance
10.1.3 Big Query File_Path = "D:\BigQuery_example\"


```

folder_path <- "D:\\BigQuery_example"
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)

```

10.2 Create .XLSX files

10.2.1 Tableau always gives me problems when I try to use anything other than .xlsx files so i find it a good practice to make both

10.2.2 Create a folder path designated for Tableau to make easier on any new user for find the correct files.

10.2.3 Tableau File_Path = "D:\\Tableau_example\\"

```

write.xlsx(NBA_Roster, file = "D:\\Tableau_example\\NBA_Roster.xlsx")
write.xlsx(NBA_Starting_Lineups, file = "D:\\Tableau_example\\NBA_Starting_Lineups.xlsx")

```

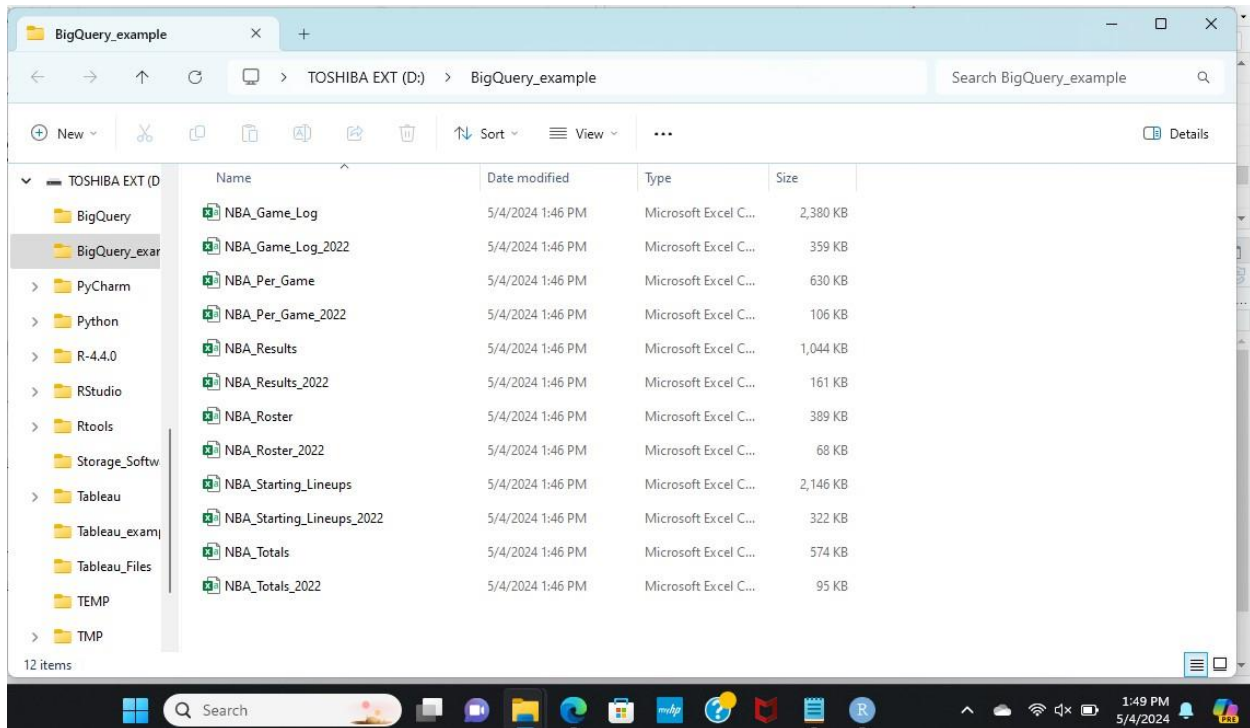


Figure 5: .CSV files after upload

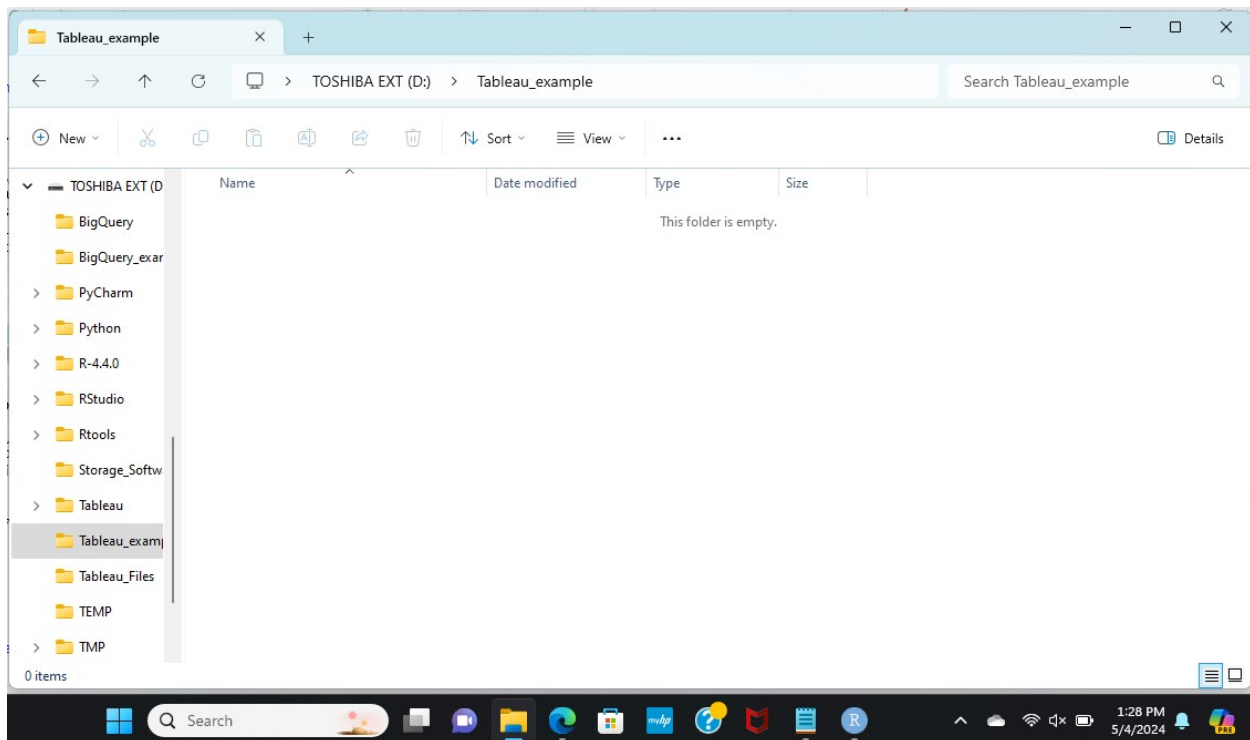


Figure 6: Tableau files before Upload

```

write.xlsx(NBA_Per_Game, file = "D:\\Tableau_example\\NBA_Per_Game.xlsx")
write.xlsx(NBA_Totals, file = "D:\\Tableau_example\\NBA_Totals.xlsx")
write.xlsx(NBA_Game_Log, file =

```

```
"D:\\Tableau_example\\NBA_Game_Log.xlsx") write.xlsx(NBA_Results,
file = "D:\\Tableau_example\\NBA_Results.xlsx")
```

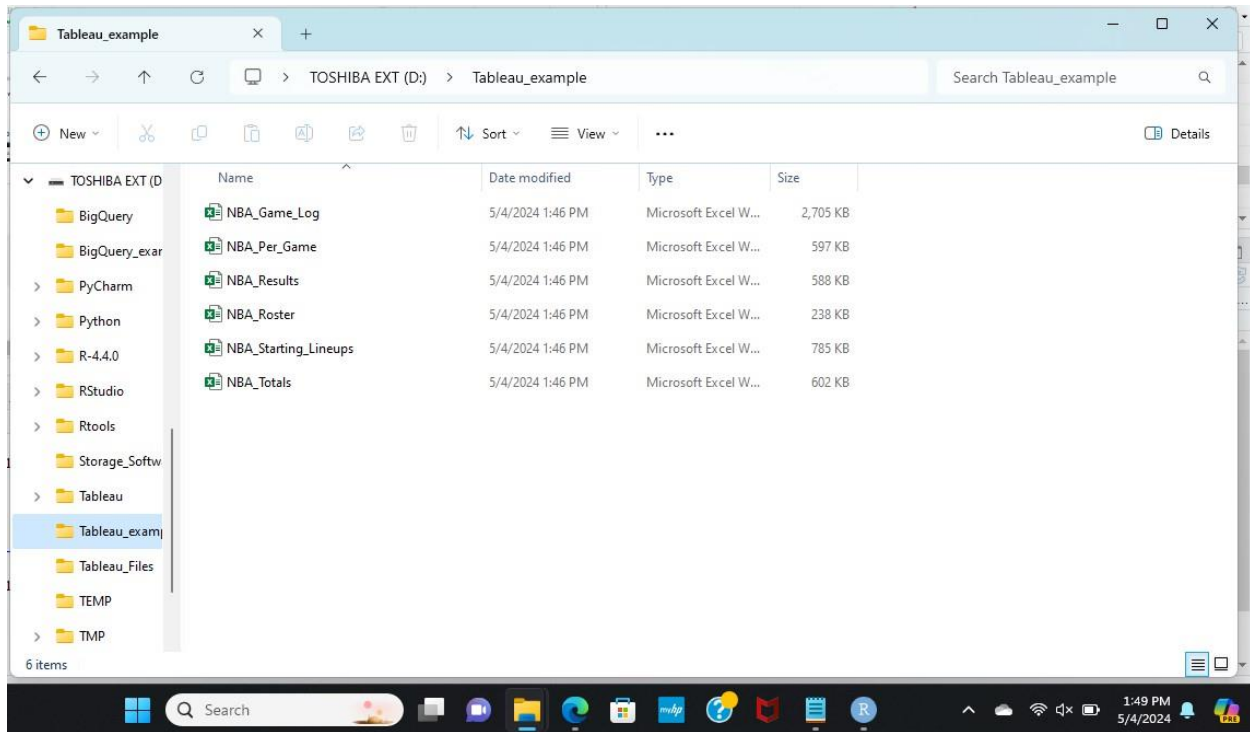


Figure 7: Tableau files after Upload

11 Conclusion

11.1 ETL

11.1.1 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.

11.2 Portfolio Project

11.2.1 Links to the entire [Basketball Data Project](#) including the [Python Web Scrape Code](#) and [Tableau Dashboards](#) can be found on my [GitHub Profile](#).