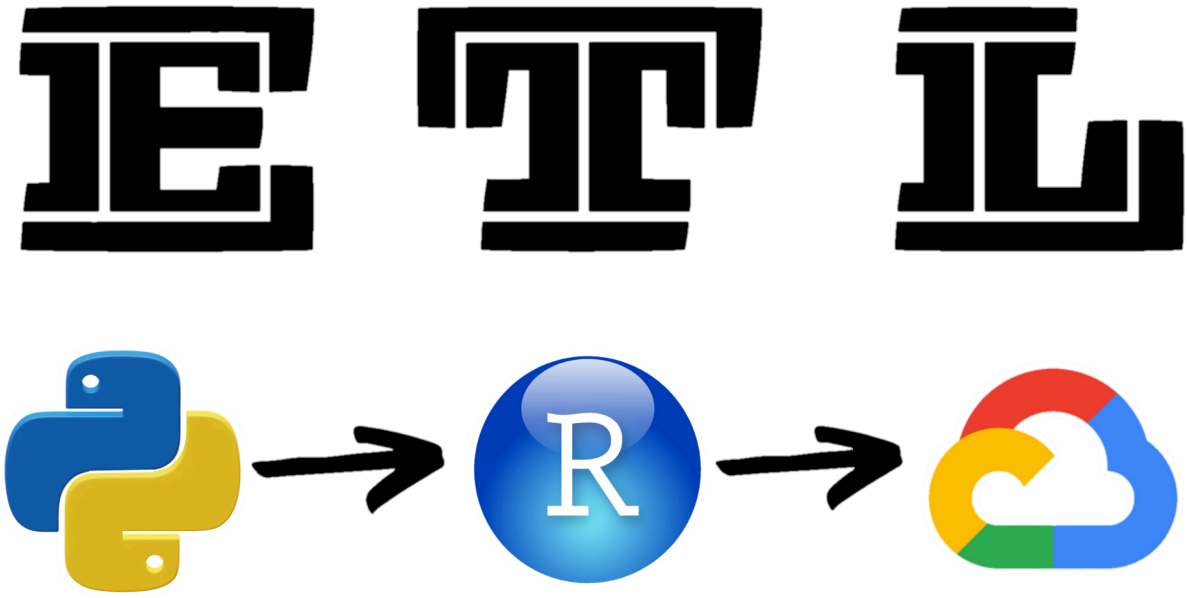


Data Warehousing (ETL)

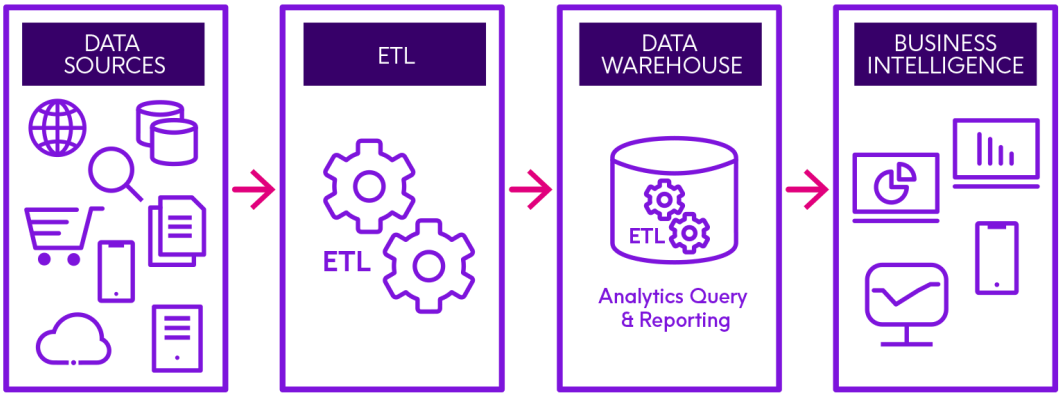
LOAD

Sawandi Kirby

2024-07-22

Contents

ETL -(Extract Transform Load)



Extract Transform Load

Definition

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:

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

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

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.

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

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

Extract

Python- Web Scrape

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

–Load

Import data

Load Required Packages

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

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

```
library(tidyverse)
```

```
library(readxl)
library(openxlsx)

library(dplyr)

library(stringr)

library(htmltools)
library(httpuv)
library(googleCloudStorageR)
library(googleAuthR)
library(bigrquery)
library(jsonlite)

library(RMySQL)

library(RODBC)

library(odbc)
library(AzureStor)
```

Import .csv files

Import the data in case you do not have it already loaded into the system

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


## Rows: 530 Columns: 24
## ── Column specification
──────────────────────────────────────────────────────────────────
## Delimiter: ","
## chr (13): Team, Country_abr, Player, FirstName, LastName, Position,
Coll...
## dbl (10): Team_Index, Height, Weight, Zip, Latitude, Longitude,
Country_...
## date (1): BirthDate
##
## 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_Per_Game                                                <-
read_csv("D:\\BigQuery\\NBA_Per_Game.csv")
```

```
## Rows: 572 Columns: 34
## ── Column specification ────────────────────────────────────────────────────
## Delimiter: ","
## chr (5): Team, Player, FirstName, LastName, Position
## dbl (29): Team_Index, Age, GamesPlayed, GamesStarted, MinutesPlayed,
Fiel...
##
## 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_Totals <- read_csv("D:\\BigQuery\\NBA_Totals.csv")
```

```
## Rows: 572 Columns: 33
## ── Column specification ────────────────────────────────────────────────────
## Delimiter: ","
## chr (5): Player, FirstName, LastName, Team, Position
## dbl (28): Team_Index, Age, TotalGamesPlayed, TotalGamesStarted,
TotalMinu...
##
## 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_Starting_Lineups                                                    <-
read_csv("D:\\BigQuery\\NBA_Starting_Lineups.csv")
```

```
## Rows: 20676 Columns: 16
## ── Column specification ────────────────────────────────────────────────────
## Delimiter: ","
## chr (5): Opponent, StartingLineup, FirstName, LastName, Team
## dbl (7): Team_Index, TeamPoints, OpponentPoints, Wins, Losses,
Player_ID...
## lgl (3): HomeAway, WinLoss, Overtime
## date (1): 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_Game_Log <- read_csv("D:\\BigQuery\\NBA_Game_Log.csv")
```

```
## Rows: 2460 Columns: 45
## ── Column specification
──────────────────────────────────────────────────────────────────────────
## Delimiter: ","
## chr (5): Opp, Opponent, OpponentStrength, Team, TeamWinLoss
## dbl (38): Team_Index, Points, OppPoints, FieldGoals, FieldGoalAttempts,
...
## lgl (1): WinLoss
## date (1): 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_Results                                              <-
read_csv("D:\\BigQuery\\NBA_Results.csv")
```

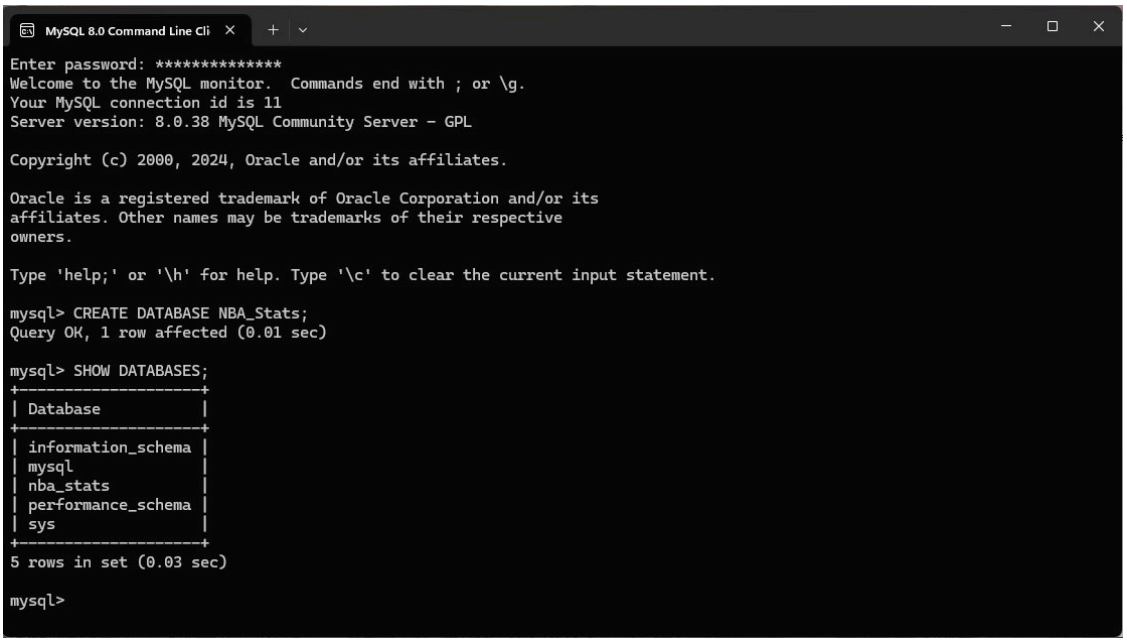
```
## Rows: 2472 Columns: 13
## ── Column specification
──────────────────────────────────────────────────────────────────────────
## Delimiter: ","
## chr (3): Opponent, Team, TeamColor
## dbl (6): Team_Index, Points, OppPoints, OpponentWins, OpponentLosses,
Ga...
## lgl (3): HomeAway, Winloss, Overtime
## date (1): 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_team_tiers <- read.csv("D:\\BigQuery\\NBA_team_tiers.csv")
Player_Performance_Long <-
read.csv("D:\\BigQuery\\Player_Performance_Long.csv"
)
```

Load Data

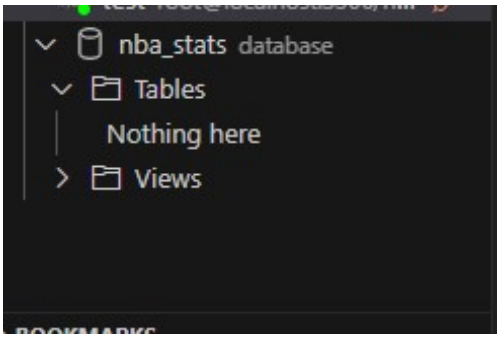
MYSQL server

There are multiple options to Uplaod your data into MYSQL server using MYSQL Workbench, VisualStudioCode and many other softwares. AS well As directly through RStudio like we have already done with Google BigQuery First we will insert the table data using SQL directly on the MYSQL server



Database_Creation

USE nba_stats;



Database Start

CREATE TABLE Roster (

```
Team VARCHAR(255),
Team_Index VARCHAR(255),
Country_abr VARCHAR(255),
Player VARCHAR(255),
FirstName VARCHAR(255),
LastName VARCHAR(255),
Position VARCHAR(255),
Height INT,
Weight INT,
BirthDate DATE,
College VARCHAR(255),
Address VARCHAR(255),
City VARCHAR(255),
State VARCHAR(255),
Zip INT,
County VARCHAR(255),
Latitude DECIMAL(10, 2),
Longitude DECIMAL(10, 2),
Country VARCHAR(255),
Country_abr3 VARCHAR(255),
Country_num INT,
Cty_Latitude DECIMAL(10, 2),
Cty_Longitude DECIMAL(10, 2),
Player_ID INT PRIMARY KEY
);
```

LOAD DATA INFILE ‘C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/NBA_Roster.csv’

INTO TABLE roster

FIELDS TERMINATED BY ‘,’

ENCLOSED BY ‘“’

LINES TERMINATED BY ‘\r\n’

IGNORE 1 ROWS;

Team	Team_Index	Country_abr	Player	FirstName	LastName
Filter...	Filter...	Filter...	Filter...	Filter...	Filter...
ATL	1	ao	bruno fernando	bruno	fernando
ATL	1	ch	clint capela	clint	capela
ATL	1	cz	vit krejci	vit	krejci
ATL	1	rs	bogdan bogdanović	bogdan	bogdanović
ATL	1	sn	mouhamed gueye	mouhamed	gueye
ATL	1	us	dejounte murray	dejounte	murray
ATL	1	us	jalen johnson	jalen	johnson
ATL	1	us	garrison mathews	garrison	mathews
ATL	1	us	saddiq bey	saddiq	bey
ATL	1	us	de'andre hunter	de'andre	hunter
ATL	1	us	trent forrest	trent	forrest
ATL	1	us	onyeka okongwu	onyeka	okongwu
ATL	1	us	trae young	trae	young
ATL	1	us	aj griffin	aj	griffin
ATL	1	us	kobe bufkin	kobe	bufkin
ATL	1	us	wesley matthews	wesley	matthews
ATL	1	us	dylan windler	dylan	windler
ATL	1	us	seth lundy	seth	lundy
BOS	2	ca	oshae brissett	oshae	brissett

CONSOLE RE-RUN QUERY EXPORT OPEN 1-50 of 530

CREATE TABLE Roster

CREATE TABLE per_game (

Team VARCHAR(255),

Team_Index INT,

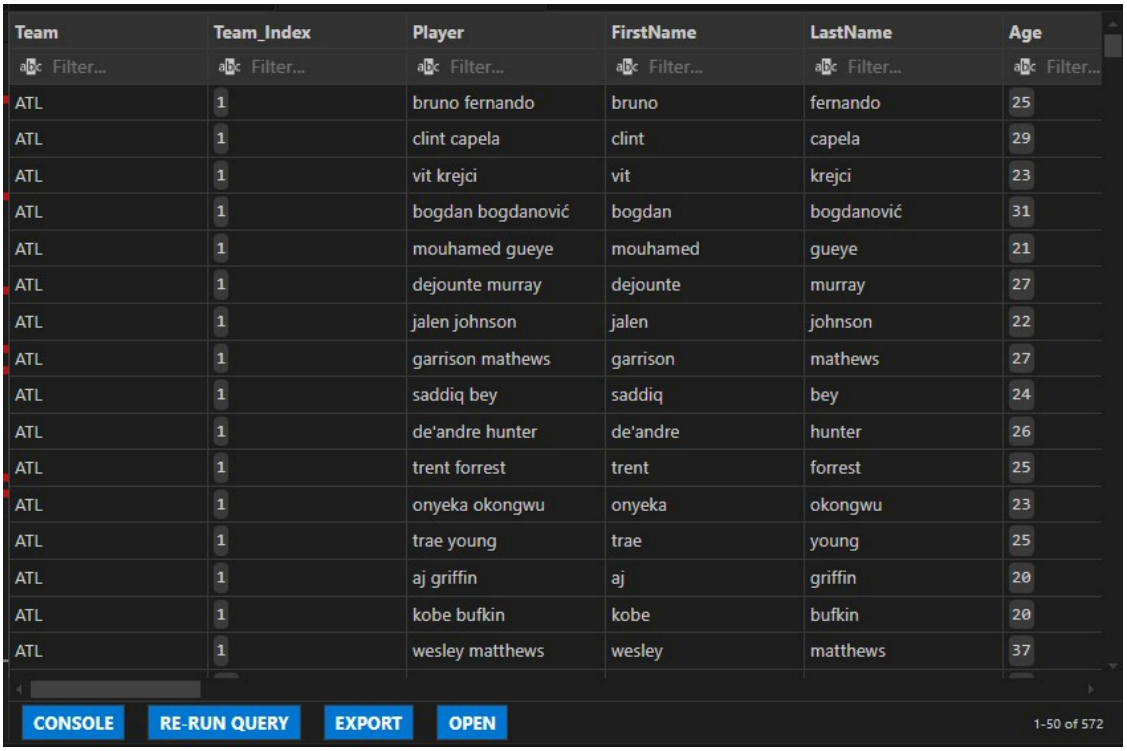
Player VARCHAR(255),

FirstName VARCHAR(255),

LastName VARCHAR(255),

```
Age INT,
GamesPlayed DECIMAL(8, 2),
GamesStarted DECIMAL(8, 2),
MinutesPlayed DECIMAL(8, 2),
FieldGoalsPerGame DECIMAL(8, 2),
FieldGoalAttemptsPerGame DECIMAL(8, 2),
FieldGoalPercentPerGame DECIMAL(8, 3),
ThreePointFieldGoalsPerGame DECIMAL(8, 2),
ThreePointFieldGoalAttemptsPerGame DECIMAL(8, 2),
ThreePointFieldGoalPercentPerGame DECIMAL(8, 3),
TwoPointFieldGoalsPerGame DECIMAL(8, 2),
TwoPointFieldGoalAttemptsPerGame DECIMAL(8, 2),
TwoPointFieldGoalPercentPerGame DECIMAL(8, 3),
EffectiveFieldGoalPercent DECIMAL(8, 3),
FreeThrowsPerGame DECIMAL(8, 2),
FreeThrowAttemptsPerGame DECIMAL(8, 2),
FreeThrowPercentPerGame DECIMAL(8, 3),
OffensiveReboundsPerGame DECIMAL(8, 2),
DefensiveReboundsPerGame DECIMAL(8, 2),
TotalReboundsPerGame DECIMAL(8, 2), AssistsPerGame DECIMAL(8,
2),
StealsPerGame DECIMAL(8, 2),
BlocksPerGame DECIMAL(8, 2),
TurnoversPerGame DECIMAL(8, 2),
PersonalFoulsPerGame DECIMAL(8, 2),
```

```
PointsPerGame DECIMAL(8, 2),
PER DECIMAL(8, 3),
Position VARCHAR(255),
Player_ID INT,
PRIMARY KEY (Player_ID)
);
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/NBA_Per_Game.csv'
INTO TABLE per_game
FIELDS TERMINATED BY ','
ENCLOSED BY ""
LINES TERMINATED BY '\r\n'
IGNORE 1 ROWS;
```



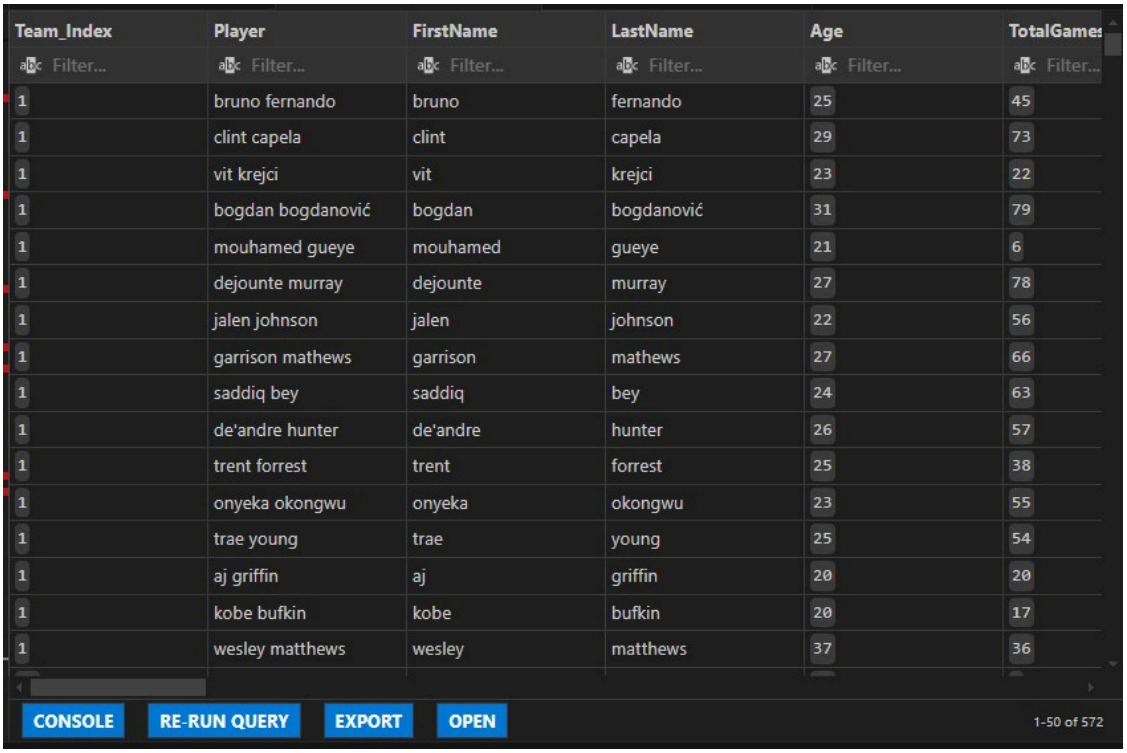
Team	Team_Index	Player	FirstName	LastName	Age
ATL	1	bruno fernando	bruno	fernando	25
ATL	1	clint capela	clint	capela	29
ATL	1	vit krejci	vit	krejci	23
ATL	1	bogdan bogdanović	bogdan	bogdanović	31
ATL	1	mouhamed gueye	mouhamed	gueye	21
ATL	1	dejounte murray	dejounte	murray	27
ATL	1	jalen johnson	jalen	johnson	22
ATL	1	garrison mathews	garrison	mathews	27
ATL	1	saddiq bey	saddiq	bey	24
ATL	1	de'andre hunter	de'andre	hunter	26
ATL	1	trent forrest	trent	forrest	25
ATL	1	onyeka okongwu	onyeka	okongwu	23
ATL	1	trae young	trae	young	25
ATL	1	aj griffin	aj	griffin	20
ATL	1	kobe bufkin	kobe	bufkin	20
ATL	1	wesley matthews	wesley	matthews	37

CONSOLE RE-RUN QUERY EXPORT OPEN 1-50 of 572

CREATE TABLE Per Game

```
CREATE TABLE totals (
Team_Index INT,
Player VARCHAR(255),
FirstName VARCHAR(255),
LastName VARCHAR(255),
Age INT,
TotalGamesPlayed INT,
TotalGamesStarted INT,
TotalMinutesPlayed DECIMAL(8, 2),
TotalFieldGoals DECIMAL(8, 2),
TotalFieldGoalAttempts DECIMAL(8, 2),
TotalFieldGoalPercent DECIMAL(8, 3),
TotalThreePointFieldGoals DECIMAL(8, 2),
TotalThreePointFieldGoalAttempts DECIMAL(8, 2),
TotalThreePointFieldGoalPercent DECIMAL(8, 3),
TotalTwoPointFieldGoals DECIMAL(8, 2),
TotalTwoPointFieldGoalAttempts DECIMAL(8, 2),
TotalTwoPointFieldGoalPercent DECIMAL(8, 3),
TotalEffectiveFieldGoalPercent DECIMAL(8, 3),
TotalFreeThrows DECIMAL(8, 2),
TotalFreeThrowAttempts DECIMAL(8, 2),
TotalFreeThrowPercent DECIMAL(8, 3),
TotalOffensiveRebounds DECIMAL(8, 2),
TotalDefensiveRebounds DECIMAL(8, 2),
TotalRebounds DECIMAL(8, 2),
```

```
TotalAssists DECIMAL(8, 2),
TotalSteals DECIMAL(8, 2),
TotalBlocks DECIMAL(8, 2),
TotalTurnovers DECIMAL(8, 2),
TotalPersonalFouls DECIMAL(8, 2),
TotalPoints DECIMAL(8, 2),
Team VARCHAR(255),
Position VARCHAR(255),
Player_ID INT,
PRIMARY KEY (Player_ID)
);
LOAD DATA INFILE ‘C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/NBA_Totals.csv’
INTO TABLE totals
FIELDS TERMINATED BY ‘,’
ENCLOSED BY ‘“’
LINES TERMINATED BY ‘\r\n’
IGNORE 1 ROWS;
```



Team_Index	Player	FirstName	LastName	Age	TotalGames
1	bruno fernando	bruno	fernando	25	45
1	clint capela	clint	capela	29	73
1	vit krejci	vit	krejci	23	22
1	bogdan bogdanović	bogdan	bogdanović	31	79
1	mouhamed gueye	mouhamed	gueye	21	6
1	dejounte murray	dejounte	murray	27	78
1	jalen johnson	jalen	johnson	22	56
1	garrison mathews	garrison	mathews	27	66
1	saddiq bey	saddiq	bey	24	63
1	de'andre hunter	de'andre	hunter	26	57
1	trent forrest	trent	forrest	25	38
1	onyeka okongwu	onyeka	okongwu	23	55
1	trae young	trae	young	25	54
1	aj griffin	aj	griffin	20	20
1	kobe bufkin	kobe	bufkin	20	17
1	wesley matthews	wesley	matthews	37	36

CONSOLE RE-RUN QUERY EXPORT OPEN 1-50 of 572

CREATE TABLE Totals

```
CREATE TABLE starting_lineups (
Team_Index INT,
Date DATE,
HomeAway TINYINT,
Opponent VARCHAR(255),
WinLoss TINYINT,
Overtime TINYINT,
TeamPoints INT,
OpponentPoints INT,
Wins INT,
Losses INT,
StartingLineup VARCHAR(255),
FirstName VARCHAR(255),
LastName VARCHAR(255),
Team VARCHAR(255),
Player_ID INT,
Game_ID INT,
id INT AUTO_INCREMENT PRIMARY KEY,
UNIQUE KEY (Player_ID, Game_ID)
);
LOAD DATA INFILE ‘C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/NBA_Starting_Lineups.csv’
INTO TABLE starting_lineups
FIELDS TERMINATED BY ‘,’
ENCLOSED BY ‘“’
```

LINES TERMINATED BY ‘\r\n’

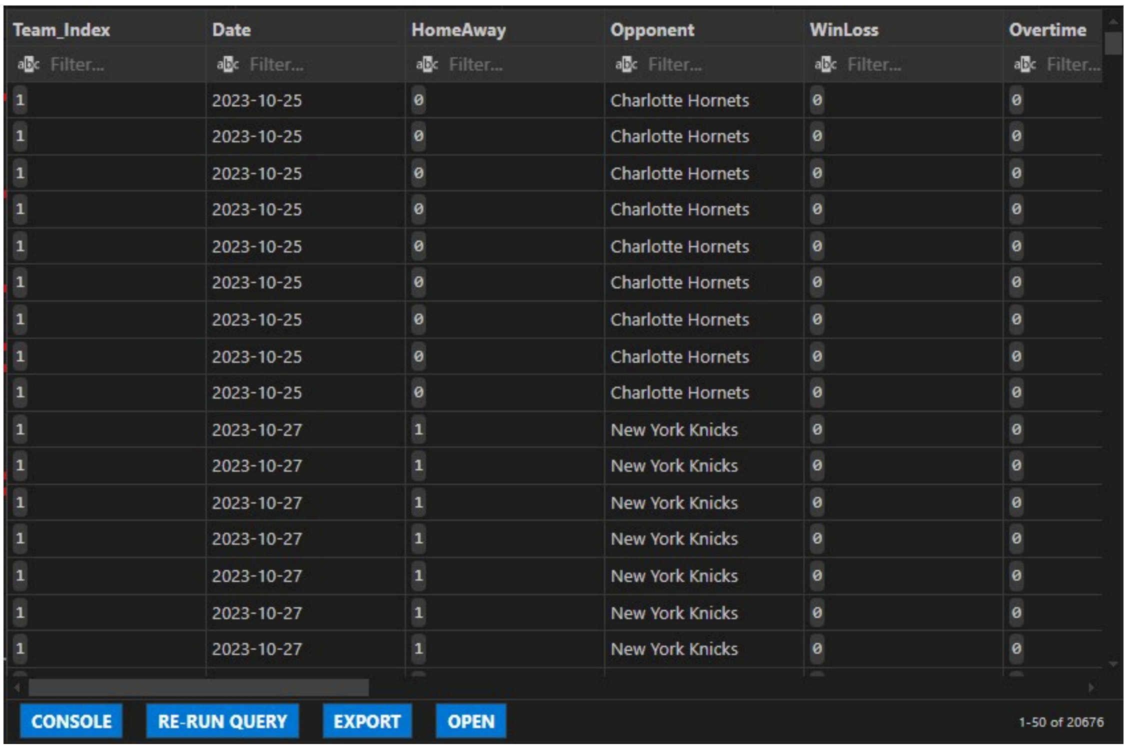
IGNORE 1 ROWS

(Team_Index, Date, @HomeAway, Opponent, @WinLoss, @Overtime, TeamPoints, OpponentPoints, Wins, Losses, StartingLineup, FirstName, LastName, Team, Player_ID, Game_ID)

SET HomeAway = (@HomeAway = ‘TRUE’),

WinLoss = (@WinLoss = ‘TRUE’),

Overtime = (@Overtime = ‘TRUE’);



Team_Index	Date	HomeAway	Opponent	WinLoss	Overtime
1	2023-10-25	0	Charlotte Hornets	0	0
1	2023-10-25	0	Charlotte Hornets	0	0
1	2023-10-25	0	Charlotte Hornets	0	0
1	2023-10-25	0	Charlotte Hornets	0	0
1	2023-10-25	0	Charlotte Hornets	0	0
1	2023-10-25	0	Charlotte Hornets	0	0
1	2023-10-25	0	Charlotte Hornets	0	0
1	2023-10-25	0	Charlotte Hornets	0	0
1	2023-10-25	0	Charlotte Hornets	0	0
1	2023-10-27	1	New York Knicks	0	0
1	2023-10-27	1	New York Knicks	0	0
1	2023-10-27	1	New York Knicks	0	0
1	2023-10-27	1	New York Knicks	0	0
1	2023-10-27	1	New York Knicks	0	0
1	2023-10-27	1	New York Knicks	0	0
1	2023-10-27	1	New York Knicks	0	0

CONSOLE RE-RUN QUERY EXPORT OPEN 1-50 of 20676

CREATE TABLE Starting Lineups

CREATE TABLE game_log (

Team_Index INT,

Date DATE,

Opp VARCHAR(255),

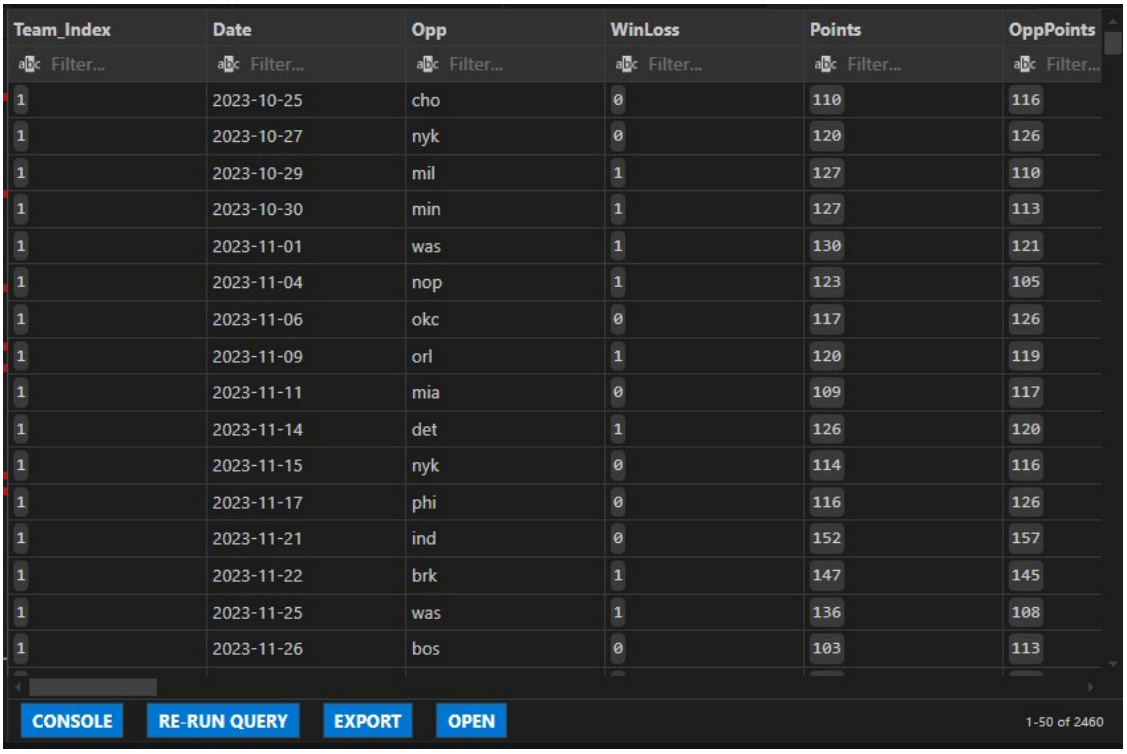
WinLoss TINYINT,

```
Points INT,
OppPoints INT,
FieldGoals INT,
FieldGoalAttempts INT,
FieldGoalPercent DECIMAL(8, 2),
ThreePoints INT,
ThreePointAttempts INT,
ThreePointPercent DECIMAL(8, 2),
FreeThrows INT,
FreeThrowAttempts INT,
FreeThrowPercent DECIMAL(8, 2),
OffensiveRebounds INT,
TotalRebounds INT,
Assists INT,
Steals INT,
Blocks INT,
Turnovers INT,
PersonalFouls INT,
OppFieldGoals INT,
OppFieldGoalAttempts INT,
OppFieldGoalPercent DECIMAL(8, 2), OppThreePoints INT,
OppThreePointAttempts INT,
OppThreePointPercent DECIMAL(8, 2),
OppFreeThrows INT,
OppFreeThrowAttempts INT,
```

```
OppFreeThrowPercent DECIMAL(8, 2),
OppOffensiveRebounds INT,
OppTotalRebounds INT,
OppAssists INT,
OppSteals INT,
OppBlocks INT,
OppTurnovers INT,
OppPersonalFouls INT,
Opponent VARCHAR(255),
OpponentWins INT,
OpponentLosses INT,
OpponentStrength VARCHAR(255),
Team VARCHAR(255),
TeamWinLoss VARCHAR(255),
Game_ID INT,
PRIMARY KEY (Game_ID),
UNIQUE KEY (Team_Index, DATE)
);
LOAD DATA INFILE ‘C:/ProgramData/MySQL/MySQL Server
8.0/Uploads/NBA_Game_Log.csv’
INTO TABLE game_log
FIELDS TERMINATED BY ‘,’
ENCLOSED BY ‘“’
LINES TERMINATED BY ‘\r\n’
IGNORE 1 ROWS
```

(Team_Index, Date, Opp, @WinLoss, Points, OppPoints, FieldGoals, FieldGoalAttempts, FieldGoalPercent, ThreePoints, ThreePointAttempts, ThreePointPercent, FreeThrows, FreeThrowAttempts, FreeThrowPercent, OffensiveRebounds, TotalRebounds, Assists, Steals, Blocks, Turnovers, PersonalFouls, OppFieldGoals, OppFieldGoalAttempts, OppFieldGoalPercent, OppThreePoints, OppThreePointAttempts, OppThreePointPercent, OppFreeThrows, OppFreeThrowAttempts, OppFreeThrowPercent, OppOffensiveRebounds, OppTotalRebounds, OppAssists, OppSteals, OppBlocks, OppTurnovers, OppPersonalFouls, Opponent, OpponentWins, OpponentLosses, OpponentStrength, Team, TeamWinLoss, Game_ID)

SET WinLoss = (@WinLoss = ‘TRUE’);



Team_Index	Date	Opp	WinLoss	Points	OppPoints
Filter...	Filter...	Filter...	Filter...	Filter...	Filter...
1	2023-10-25	cho	0	110	116
1	2023-10-27	nyk	0	120	126
1	2023-10-29	mil	1	127	110
1	2023-10-30	min	1	127	113
1	2023-11-01	was	1	130	121
1	2023-11-04	nop	1	123	105
1	2023-11-06	okc	0	117	126
1	2023-11-09	orl	1	120	119
1	2023-11-11	mia	0	109	117
1	2023-11-14	det	1	126	120
1	2023-11-15	nyk	0	114	116
1	2023-11-17	phi	0	116	126
1	2023-11-21	ind	0	152	157
1	2023-11-22	brk	1	147	145
1	2023-11-25	was	1	136	108
1	2023-11-26	bos	0	103	113

CONSOLE RE-RUN QUERY EXPORT OPEN 1-50 of 2460

CREATE TABLE Game Log

```
CREATE TABLE
                    results (
Team_Index INT,
Date DATE,
HomeAway TINYINT,
Opponent VARCHAR(255),
Winloss TINYINT,
Overtime TINYINT,
Points INT,
OppPoints INT,
OpponentWins INT,
OpponentLosses INT,
Team VARCHAR(255),
TeamColor VARCHAR(255),
Game_ID INT,
PRIMARY KEY (Game_ID),
UNIQUE KEY (Team_Index, DATE)
);
LOAD DATA INFILE ‘C:/ProgramData/MySQL/MySQL Server
8.0/Uploads/NBA_Results.csv’
INTO TABLE results
FIELDS TERMINATED BY ‘,’
ENCLOSED BY ‘“’
LINES TERMINATED BY ‘\r\n’
IGNORE 1 ROWS
```

(Team_Index, Date, @HomeAway, Opponent, @Winloss, @Overtime, Points, OppPoints, OpponentWins, OpponentLosses, Team, TeamColor, Game_ID)

SET HomeAway = (@HomeAway = ‘TRUE’),

Winloss = (@Winloss = ‘TRUE’),

Overtime = (@Overtime = ‘TRUE’);

Team_Index	Date	HomeAway	Opponent	Winloss	Overtime
Filter...	Filter...	Filter...	Filter...	Filter...	Filte
1	2023-10-25	0	charlotte hornets	0	0
1	2023-10-27	1	new york knicks	0	0
1	2023-10-29	0	milwaukee bucks	1	0
1	2023-10-30	1	minnesota timberwolves	1	0
1	2023-11-01	1	washington wizards	1	0
1	2023-11-04	0	new orleans pelicans	1	0
1	2023-11-06	0	oklahoma city thunder	0	0
1	2023-11-09	0	orlando magic	1	0
1	2023-11-11	1	miami heat	0	0
1	2023-11-14	0	detroit pistons	1	0
1	2023-11-15	1	new york knicks	0	0
1	2023-11-17	1	philadelphia 76ers	0	0
1	2023-11-21	1	indiana pacers	0	0
1	2023-11-22	1	brooklyn nets	1	1
1	2023-11-25	0	washington wizards	1	0
1	2023-11-26	0	boston celtics	0	0

CONSOLE RE-RUN QUERY EXPORT OPEN 1-50 of 2472

CREATE TABLE Results

team_tiers (

Team_Index INT,

TeamWins INT,

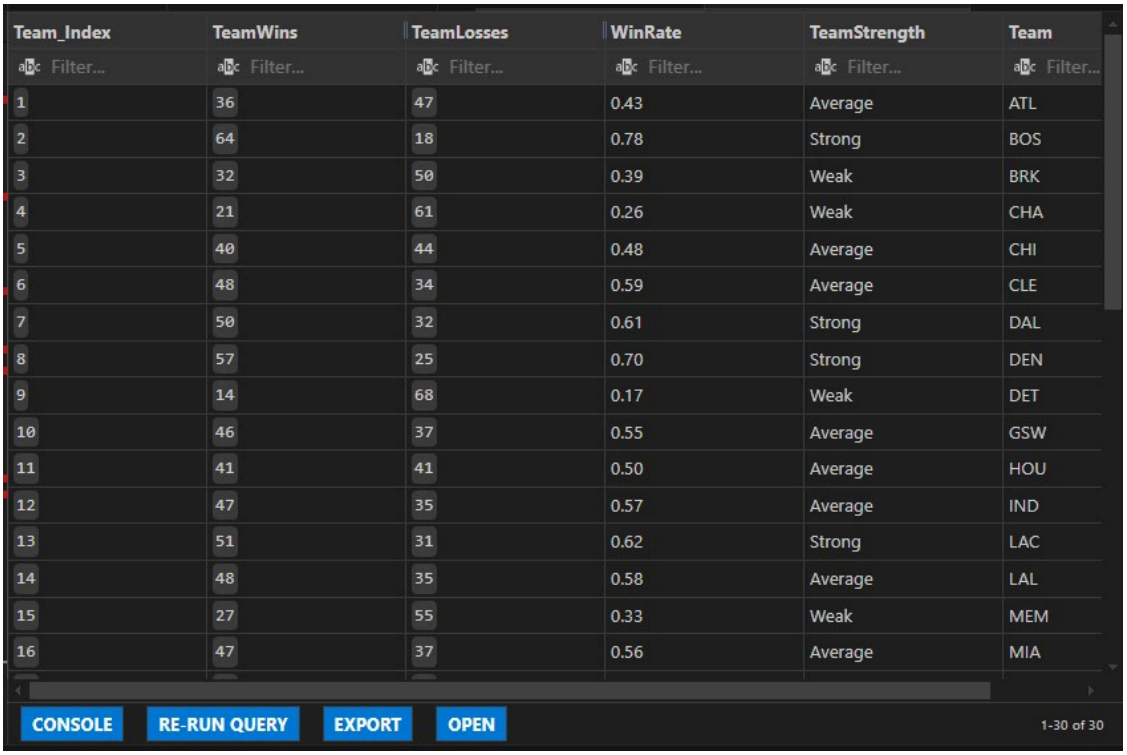
TeamLosses INT,

WinRate DECIMAL(8, 2),

TeamStrength VARCHAR(255),

Team VARCHAR(255),

```
CREATE TABLE

PRIMARY KEY (Team_Index)

);

LOAD DATA INFILE ‘C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/NBA_Team_Tiers.csv’

INTO TABLE team_tiers

FIELDS TERMINATED BY ‘,’

ENCLOSED BY ‘“’

LINES TERMINATED BY ‘\r\n’

IGNORE 1 ROWS;
```



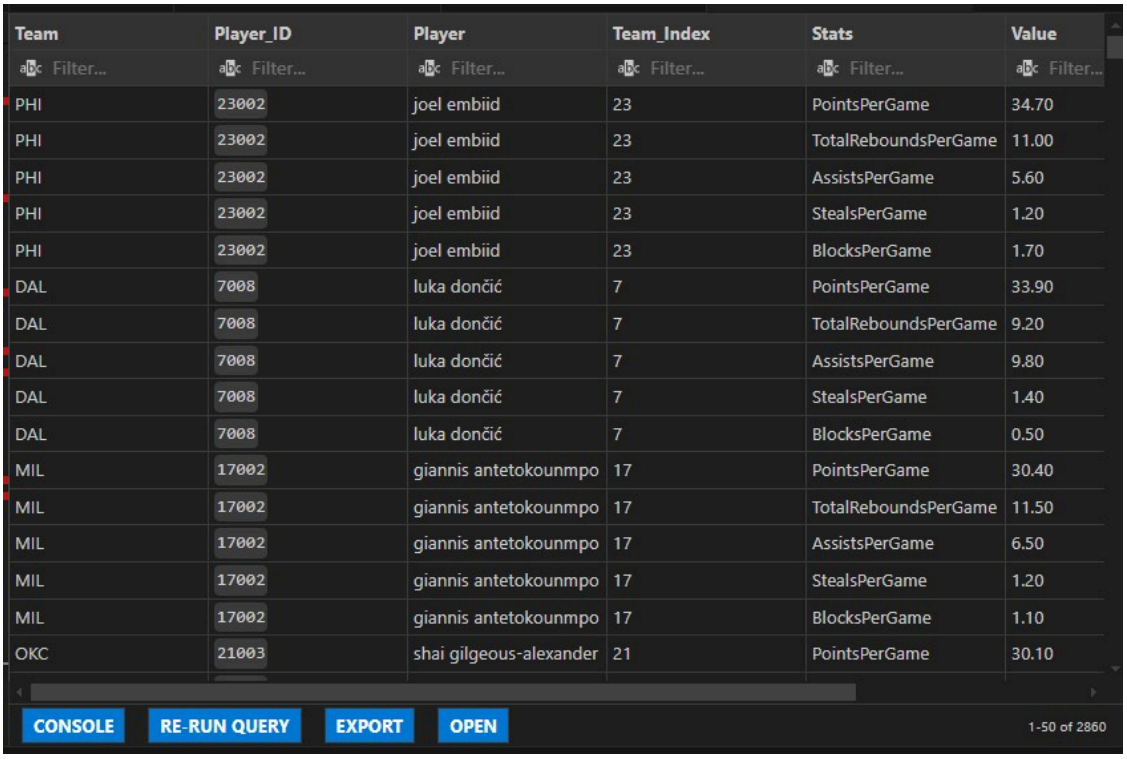
Team_Index	TeamWins	TeamLosses	WinRate	TeamStrength	Team
1	36	47	0.43	Average	ATL
2	64	18	0.78	Strong	BOS
3	32	50	0.39	Weak	BRK
4	21	61	0.26	Weak	CHA
5	40	44	0.48	Average	CHI
6	48	34	0.59	Average	CLE
7	50	32	0.61	Strong	DAL
8	57	25	0.70	Strong	DEN
9	14	68	0.17	Weak	DET
10	46	37	0.55	Average	GSW
11	41	41	0.50	Average	HOU
12	47	35	0.57	Average	IND
13	51	31	0.62	Strong	LAC
14	48	35	0.58	Average	LAL
15	27	55	0.33	Weak	MEM
16	47	37	0.56	Average	MIA

CONSOLE RE-RUN QUERY EXPORT OPEN 1-30 of 30

CREATE TABLE Team Tiers

```
                player_performance (
Team VARCHAR(255),
Player_ID INT,
Player VARCHAR(255),
Team_Index VARCHAR(255),
Stats VARCHAR(255), Value DECIMAL(8,
2), id INT AUTO_INCREMENT PRIMARY
KEY,
UNIQUE KEY (Player_ID, Stats)
);
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server
8.0/Uploads/Player_Performance_Long.csv'
INTO TABLE player_performance
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 ROWS
(Team, Player_ID, Player, Team_Index, Stats, Value);
```

CREATE TABLE



Team	Player_ID	Player	Team_Index	Stats	Value
PHI	23002	joel embiid	23	PointsPerGame	34.70
PHI	23002	joel embiid	23	TotalReboundsPerGame	11.00
PHI	23002	joel embiid	23	AssistsPerGame	5.60
PHI	23002	joel embiid	23	StealsPerGame	1.20
PHI	23002	joel embiid	23	BlocksPerGame	1.70
DAL	7008	luka dončić	7	PointsPerGame	33.90
DAL	7008	luka dončić	7	TotalReboundsPerGame	9.20
DAL	7008	luka dončić	7	AssistsPerGame	9.80
DAL	7008	luka dončić	7	StealsPerGame	1.40
DAL	7008	luka dončić	7	BlocksPerGame	0.50
MIL	17002	giannis antetokounmpo	17	PointsPerGame	30.40
MIL	17002	giannis antetokounmpo	17	TotalReboundsPerGame	11.50
MIL	17002	giannis antetokounmpo	17	AssistsPerGame	6.50
MIL	17002	giannis antetokounmpo	17	StealsPerGame	1.20
MIL	17002	giannis antetokounmpo	17	BlocksPerGame	1.10
OKC	21003	shai gilgeous-alexander	21	PointsPerGame	30.10

CONSOLE RE-RUN QUERY EXPORT OPEN 1-50 of 2860

CREATE TABLE Player Performance

Load and update tables directly from RStudio

MySQL database connection settings

Connect to your MySQL database

```
mysql_settings <- list(
  dbname = Sys.getenv("MYSQL_DBNAME"),
  host = Sys.getenv("MYSQL_HOST"),
  port =
  as.integer(Sys.getenv("MYSQL_PORT")),
  user = Sys.getenv("MYSQL_USER"),

  password =
  Sys.getenv("MYSQL_PASSWORD")
)


con <- dbConnect(RMySQL::MySQL(),
                 dbname = mysql_settings$dbname,
                 host = mysql_settings$host,
```

```
                port = mysql_settings$port,
                user = mysql_settings$user,
                password = mysql_settings$password,
                local_infile = 1)
```

Create the respective tables

Then close the database connection

```
dbWriteTable(con, "roster", NBA_Roster, row.names = FALSE, overwrite = TRUE)
```

```
## [1] TRUE
```

```
dbWriteTable(con, "per_game", NBA_Per_Game, row.names = FALSE, overwrite =
TRUE)
```

```
## [1] TRUE
```

```
dbWriteTable(con, "Totals", NBA_Totals, row.names = FALSE, overwrite =
TRUE)
```

```
## [1] TRUE
```

```
dbWriteTable(con, "starting_lineups", NBA_Starting_Lineups, row.names =
FALSE, overwrite = TRUE)
```

```
## [1] TRUE
```

```
dbWriteTable(con, "game_log", NBA_Game_Log, row.names = FALSE, overwrite =
TRUE)
```

```
## [1] TRUE
```

```
dbWriteTable(con, "results", NBA_Results, row.names = FALSE, overwrite
=
TRUE
)
```

```
## [1] TRUE
```

```
dbWriteTable(con, "team_tiers", NBA_team_tiers, row.names = FALSE, overwrite
= TRUE)
```

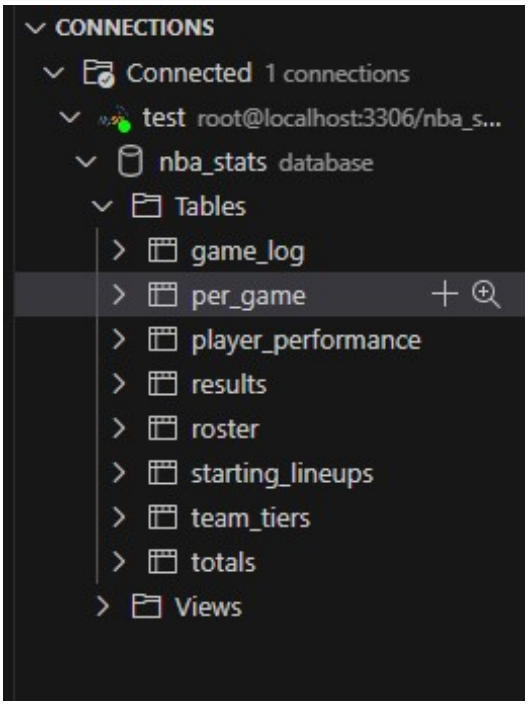
```
## [1] TRUE
```

```
dbWriteTable(con, "player_performance", Player_Performance_Long, row.names =
FALSE, overwrite = TRUE)
```

```
## [1] TRUE
```

```
dbDisconnect(con)
```

```
## [1] TRUE
```



Database Complete

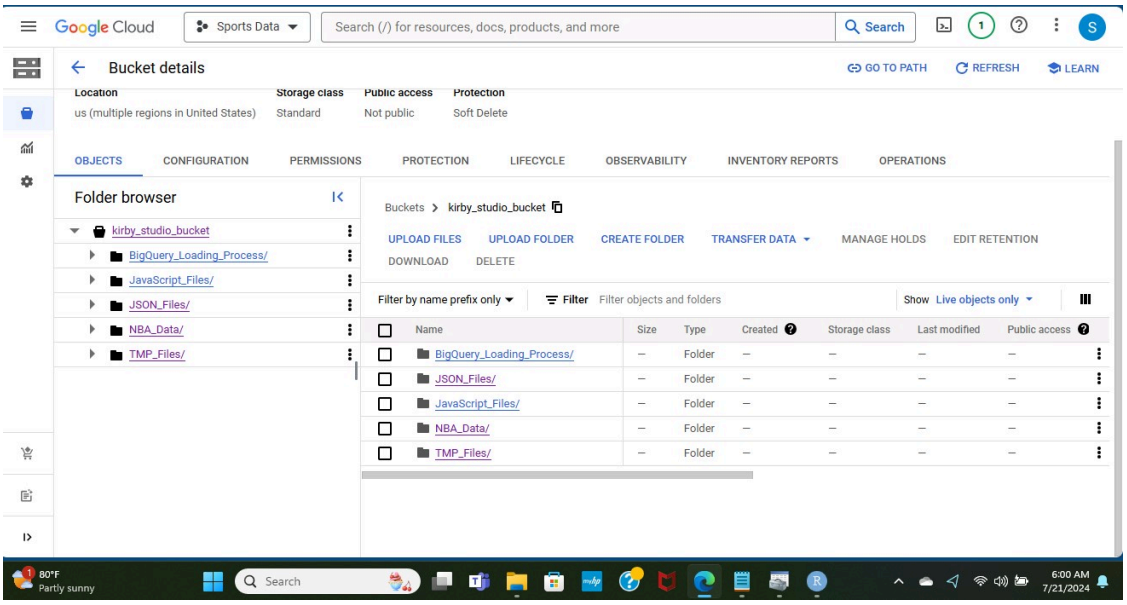
Google

Google Cloud Storage

Let's save our Data to `Google Cloud Storage` to have an extra layer of storage.

Load environment variables. Use `Sys.getenv` "System get environment" to accomplish this.

Authenticate with Google Cloud Storage using a `service account key file`.



Google Cloud Storage before upload

```
gcs_key_file <- Sys.getenv("GCS_KEY_FILE")


gcs_auth(gcs_key_file)
```

Upload the files.

```
gcs_upload(file = "D:\\BigQuery_example\\NBA_Roster.csv", bucket =
"kirby_studio_bucket", predefinedAcl = "bucketLevel")

## ==Google Cloud Storage Object==
## Name:                D:\\BigQuery_example\\NBA_Roster.csv
## Type:                text/csv
## Size:                120 Kb
## Media URL:
https://www.googleapis.com/download/storage/v1/b/kirby_studio_bucket/o/D:%5C%
```

```
5CBigQuery_example%5C%5CNBA_Roster.csv?generation=1721822745471448&alt=media
## Download URL:
https://storage.cloud.google.com/kirby_studio_bucket/D%3A%5C%5CBigQuery_examp

le%5C%5CNBA_Roster.csv
## Public Download URL:
https://storage.googleapis.com/kirby_studio_bucket/D%3A%5C%5CBigQuery_example
%5C%5CNBA_Roster.csv
## Bucket:              kirby_studio_bucket
## ID:
kirby_studio_bucket/D:\\BigQuery_example\\NBA_Roster.csv/1721822745471448
## MD5 Hash:            h+B98q2LahzwwiR45qv86g==
## Class:               STANDARD
## Created:             2024-07-24 12:05:45
## Updated:             2024-07-24 12:05:45
## Generation:          1721822745471448
## Meta Generation:     1
## eTag:                CNjT8MjRv4cDEAE=
## crc32c:              RzZ+iA==
```

```
gcs_upload(file = "D:\\BigQuery_example\\NBA_Starting_Lineups.csv", bucket =
"kirby_studio_bucket", predefinedAcl = "bucketLevel")
```

```
## ==Google Cloud Storage Object==
## Name:                D:\\BigQuery_example\\NBA_Starting_Lineups.csv
## Type:                text/csv
## Size:                2.2 Mb
## Media URL:
https://www.googleapis.com/download/storage/v1/b/kirby_studio_bucket/o/D:%5C%
5CBigQuery_example%5C%5CNBA_Starting_Lineups.csv?generation=1721822746501639&
alt=media
## Download URL:
https://storage.cloud.google.com/kirby_studio_bucket/D%3A%5C%5CBigQuery_examp
le%5C%5CNBA_Starting_Lineups.csv
## Public Download URL:
https://storage.googleapis.com/kirby_studio_bucket/D%3A%5C%5CBigQuery_example
%5C%5CNBA_Starting_Lineups.csv
## Bucket:              kirby_studio_bucket
## ID:
kirby_studio_bucket/D:\\BigQuery_example\\NBA_Starting_Lineups.csv/1721822746
501639
## MD5 Hash:            sFoC1/kpaHaZvxW16RaK+w==
```

```
## Class:                STANDARD
## Created:              2024-07-24 12:05:46
## Updated:              2024-07-24 12:05:46
## Generation:           1721822746501639
## Meta Generation:      1
## eTag:                 CIfEr8nRv4cDEAE=
## crc32c:               P8zbRw==
```

```
gcs_upload(file = "D:\\BigQuery_example\\NBA_Per_Game.csv", bucket =

"kirby_studio_bucket",        predefinedAcl        =
"bucketLevel")


## ==Google Cloud Storage Object==
## Name:                 D:\\BigQuery_example\\NBA_Per_Game.csv
```

```
## Type:                 text/csv
## Size:                 97.4 Kb
## Media URL:
https://www.googleapis.com/download/storage/v1/b/kirby_studio_bucket/o/D:%5C%
5CBigQuery_example%5C%5CNBA_Per_Game.csv?generation=1721822747212862&alt=medi
a
## Download URL:
https://storage.cloud.google.com/kirby_studio_bucket/D%3A%5C%5CBigQuery_examp
le%5C%5CNBA_Per_Game.csv
## Public Download URL:
https://storage.googleapis.com/kirby_studio_bucket/D%3A%5C%5CBigQuery_example
%5C%5CNBA_Per_Game.csv
## Bucket:               kirby_studio_bucket
## ID:
kirby_studio_bucket/D:\\BigQuery_example\\NBA_Per_Game.csv/1721822747212862
## MD5 Hash:             jhl6Y0PewX8IStbCJnqfcQ==
## Class:                STANDARD
## Created:              2024-07-24 12:05:47
## Updated:              2024-07-24 12:05:47
## Generation:           1721822747212862
## Meta Generation:      1
## eTag:                 CL742snRv4cDEAE=
## crc32c:               57NYtQ==
```

```
gcs_upload(file = "D:\\BigQuery_example\\NBA_Totals.csv", bucket =
```

```
"kirby_studio_bucket", predefinedAcl = "bucketLevel")
```

```
## ==Google Cloud Storage Object==
## Name:                D:\\BigQuery_example\\NBA_Totals.csv
## Type:                text/csv
## Size:                83.3 Kb
## Media URL:
https://www.googleapis.com/download/storage/v1/b/kirby_studio_bucket/o/D:%5C%
5CBigQuery_example%5C%5CNBA_Totals.csv?generation=1721822748101276&alt=media
## Download URL:
https://storage.cloud.google.com/kirby_studio_bucket/D%3A%5C%5CBigQuery_examp
le%5C%5CNBA_Totals.csv
## Public Download URL:
https://storage.googleapis.com/kirby_studio_bucket/D%3A%5C%5CBigQuery_example
%5C%5CNBA_Totals.csv
## Bucket:              kirby_studio_bucket
## ID:
kirby_studio_bucket/D:\\BigQuery_example\\NBA_Totals.csv/1721822748101276
## MD5 Hash:            HvJdv1APOCNt5yVzXCxrvw==
## Class:               STANDARD
## Created:             2024-07-24 12:05:48
## Updated:             2024-07-24 12:05:48
## Generation:          1721822748101276
## Meta Generation:     1
```

```
## eTag:                CJyVkcrRv4cDEAE=
## crc32c:              XXX/hA==
```

```
gcs_upload(file = "D:\\BigQuery_example\\NBA_Game_Log.csv", bucket =

"kirby_studio_bucket",        predefinedAcl        =
"bucketLevel")
```

```
## ==Google Cloud Storage Object==
## Name:                D:\\BigQuery_example\\NBA_Game_Log.csv
## Type:                text/csv
## Size:                472 Kb
## Media URL:
https://www.googleapis.com/download/storage/v1/b/kirby_studio_bucket/o/D:%5C%
5CBigQuery_example%5C%5CNBA_Game_Log.csv?generation=1721822749204809&alt=medi
a
```

```
## Download URL:
https://storage.cloud.google.com/kirby_studio_bucket/D%3A%5C%5CBigQuery_examp
le%5C%5CNBA_Game_Log.csv
## Public Download URL:
https://storage.googleapis.com/kirby_studio_bucket/D%3A%5C%5CBigQuery_example
%5C%5CNBA_Game_Log.csv
## Bucket:              kirby_studio_bucket
## ID:
kirby_studio_bucket/D:\\BigQuery_example\\NBA_Game_Log.csv/1721822749204809
## MD5 Hash:            9wT3ogxtJtMH/xxBs3Va7A==
## Class:               STANDARD
## Created:             2024-07-24 12:05:49
## Updated:             2024-07-24 12:05:49
## Generation:          1721822749204809
## Meta Generation:     1
## eTag:                CMnC1MrRv4cDEAE=
## crc32c:              Q7Q1fQ==
```

```
gcs_upload(file = "D:\\BigQuery_example\\NBA_Results.csv", bucket =

"kirby_studio_bucket", predefinedAcl = "bucketLevel")
```

```
## ==Google Cloud Storage Object==
## Name:                D:\\BigQuery_example\\NBA_Results.csv
## Type:                text/csv
## Size:                208.2 Kb
## Media URL:
https://www.googleapis.com/download/storage/v1/b/kirby_studio_bucket/o/D:%5C%
5CBigQuery_example%5C%5CNBA_Results.csv?generation=1721822749974765&alt=media
## Download URL:
https://storage.cloud.google.com/kirby_studio_bucket/D%3A%5C%5CBigQuery_examp
le%5C%5CNBA_Results.csv
## Public Download URL:
https://storage.googleapis.com/kirby_studio_bucket/D%3A%5C%5CBigQuery_example
%5C%5CNBA_Results.csv
## Bucket:              kirby_studio_bucket
## ID:
kirby_studio_bucket/D:\\BigQuery_example\\NBA_Results.csv/1721822749974765

## MD5 Hash:            5+nN/a8TuwAI6UBqKcHD3Q==
## Class:               STANDARD
## Created:             2024-07-24 12:05:50
```

```
## Updated:              2024-07-24 12:05:50
## Generation:           1721822749974765
## Meta Generation:      1
## eTag:                 CO3Bg8vRv4cDEAE=
## crc32c:               jOBkTg==
```

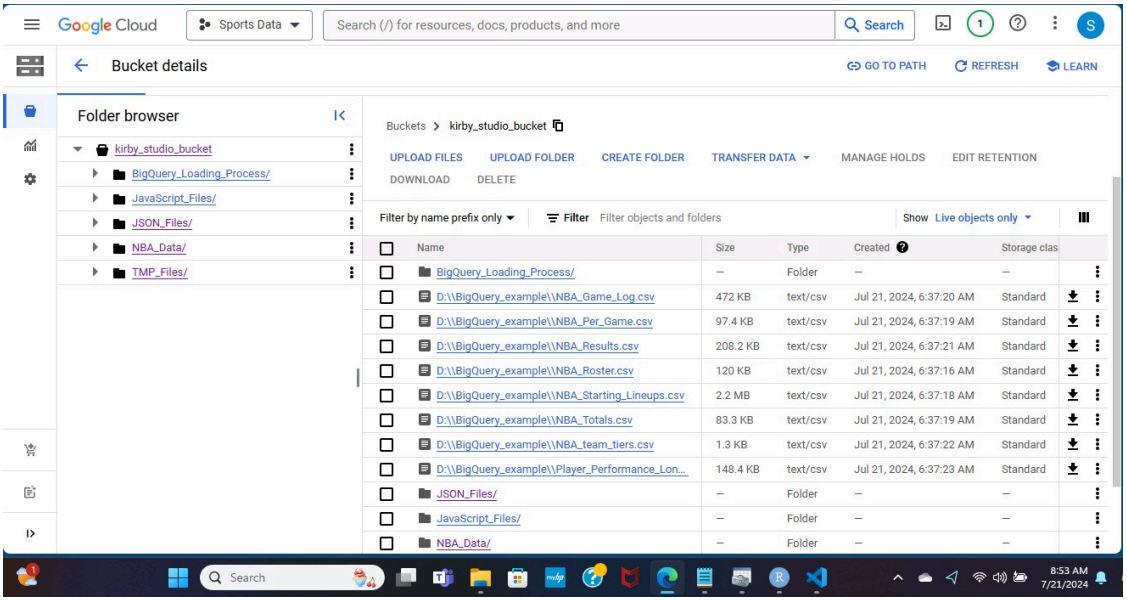
```
gcs_upload(file = "D:\\BigQuery_example\\NBA_team_tiers.csv", bucket =
"kirby_studio_bucket", predefinedAcl = "bucketLevel")
```

```
## ==Google Cloud Storage Object==
## Name:                 D:\\BigQuery_example\\NBA_team_tiers.csv
## Type:                 text/csv
## Size:                 1.3 Kb
## Media URL:
https://www.googleapis.com/download/storage/v1/b/kirby_studio_bucket/o/D:%5C%
5CBigQuery_example%5C%5CNBA_team_tiers.csv?generation=1721822750827695&alt=me
dia
## Download URL:
https://storage.cloud.google.com/kirby_studio_bucket/D%3A%5C%5CBigQuery_examp
le%5C%5CNBA_team_tiers.csv
## Public Download URL:
https://storage.googleapis.com/kirby_studio_bucket/D%3A%5C%5CBigQuery_example
%5C%5CNBA_team_tiers.csv
## Bucket:               kirby_studio_bucket
## ID:
kirby_studio_bucket/D:\\BigQuery_example\\NBA_team_tiers.csv/1721822750827695
## MD5 Hash:             v75pQvFsdV79hNKg28t7HQ==
## Class:                STANDARD
## Created:              2024-07-24 12:05:50
## Updated:              2024-07-24 12:05:50
## Generation:           1721822750827695
## Meta Generation:      1
## eTag:                 CK/Jt8vRv4cDEAE=
## crc32c:               cT4J8A==
```

```
gcs_upload(file = "D:\\BigQuery_example\\Player_Performance_Long.csv", bucket
= "kirby_studio_bucket", predefinedAcl = "bucketLevel")
```

```
## ==Google Cloud Storage Object==
## Name:                 D:\\BigQuery_example\\Player_Performance_Long.csv
## Type:                 text/csv
```

```
## Size:                148.4 Kb
## Media URL:
https://www.googleapis.com/download/storage/v1/b/kirby_studio_bucket/o/D:%5C%
5CBigQuery_example%5C%5CPlayer_Performance_Long.csv?generation=17218227515458
67&alt=media
## Download URL:
https://storage.cloud.google.com/kirby_studio_bucket/D%3A%5C%5CBigQuery_examp
le%5C%5CPlayer_Performance_Long.csv
## Public Download URL:
https://storage.googleapis.com/kirby_studio_bucket/D%3A%5C%5CBigQuery_example
%5C%5CPlayer_Performance_Long.csv
## Bucket:              kirby_studio_bucket
## ID:
kirby_studio_bucket/D:\\BigQuery_example\\Player_Performance_Long.csv/1721822
751545867
## MD5 Hash:            bTGQa89DFVU9jz2xpZmY7Q==
## Class:               STANDARD
## Created:             2024-07-24 12:05:51
## Updated:             2024-07-24 12:05:51
## Generation:          1721822751545867
## Meta Generation:     1
## eTag:                CIu048vRv4cDEAE=
## crc32c:              wBpT1w==
```



Google Cloud Storage after upload

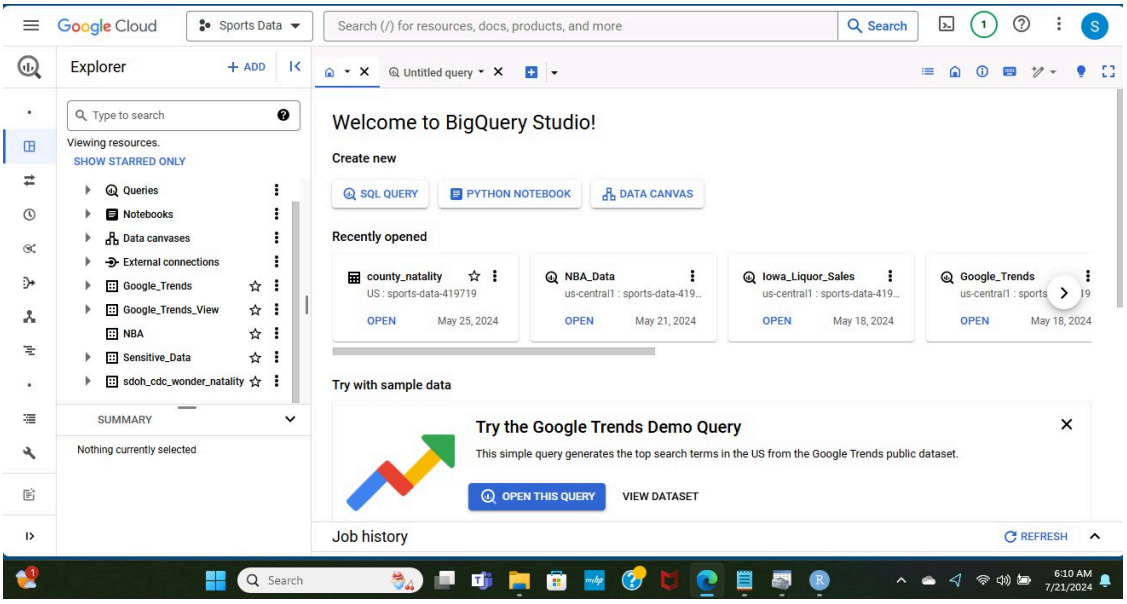
Google BigQuery

Finally we will load our data frames into tables on `Google BigQuery`. To allow us to store our data as it grows, we can query for smaller sample data sets.

This will benefit in our `ETL` process saving Time & Money!

First, authenticate with the bigquery package.

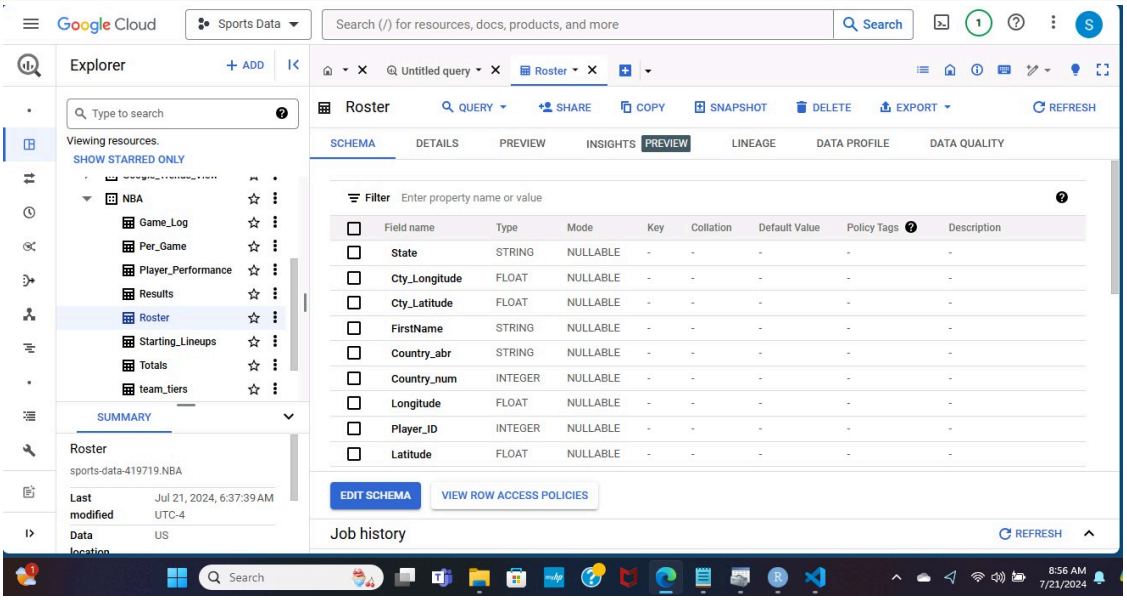
Then, upload data to BigQuery.



Google BigQuery before upload

```
bq_table_upload("sports-data-419719.NBA.Roster", NBA_Roster,
create_disposition = "CREATE_IF_NEEDED")

## ! Using an auto-discovered, cached token.

## To suppress this message, modify your code or options to clearly consent
## to the use of a cached token.

## See gargle's "Non-interactive auth" vignette for more details:

## <https://gargle.r-lib.org/articles/non-interactive-auth.html>

## ℹ The bigrquery package is using a cached token for
bq_auth                              <-
Sys.getenv("GCS_KEY_FILE")


'kirbykirb32@gmail.com'.
```

```
## Auto-refreshing stale OAuth token.
bq_table_upload("sports-data-419719.NBA.Per_Game", NBA_Per_Game,
create_disposition                    =
"CREATE_IF_NEEDED")


bq_table_upload("sports-data-419719.NBA.Totals", NBA_Totals,
create_disposition = "CREATE_IF_NEEDED")


bq_table_upload("sports-data-419719.NBA.Starting_Lineups",
NBA_Starting_Lineups, create_disposition = "CREATE_IF_NEEDED")


bq_table_upload("sports-data-419719.NBA.Game_Log", NBA_Game_Log,
create_disposition = "CREATE_IF_NEEDED")


bq_table_upload("sports-data-419719.NBA.Results", NBA_Results,
create_disposition = "CREATE_IF_NEEDED")


bq_table_upload("sports-data-419719.NBA.team_tiers", NBA_team_tiers,
create_disposition = "CREATE_IF_NEEDED")

bq_table_upload("sports-data-419719.NBA.Player_Performance",
Player_Performance_Long, create_disposition = "CREATE_IF_NEEDED")
```



Google BigQuery after upload

Microsoft AZURE

Microsoft Azure Storage

Set your Azure Storage account name and key.

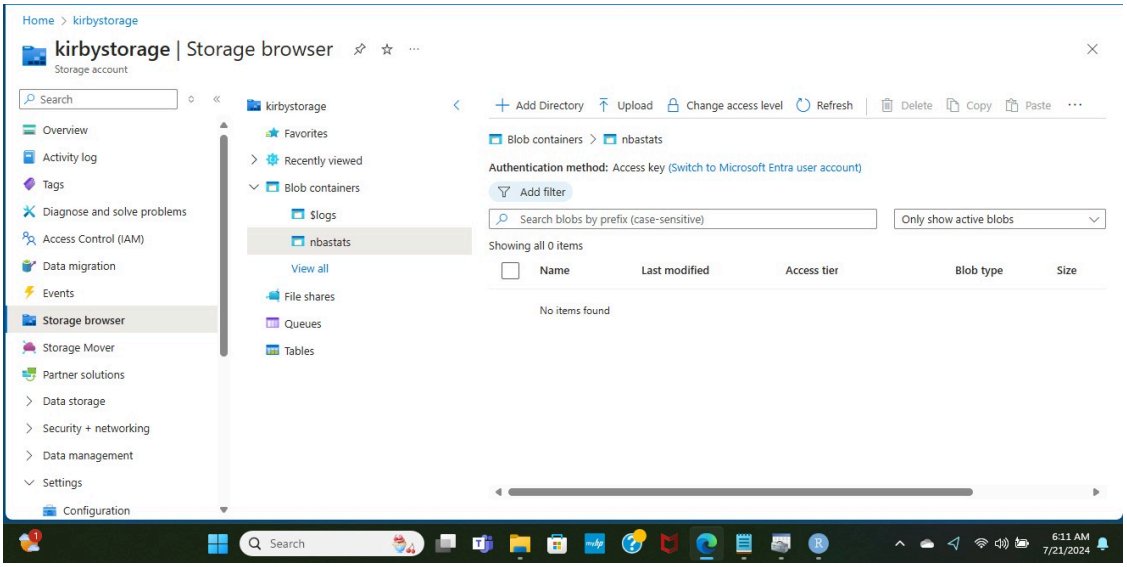
Create the storage endpoint.

Connect to the blob container.

List of data frames and their corresponding file names.

Directory to save temporary CSV files.

Loop through each data frame, save as CSV, and upload to Azure Blob Storage.



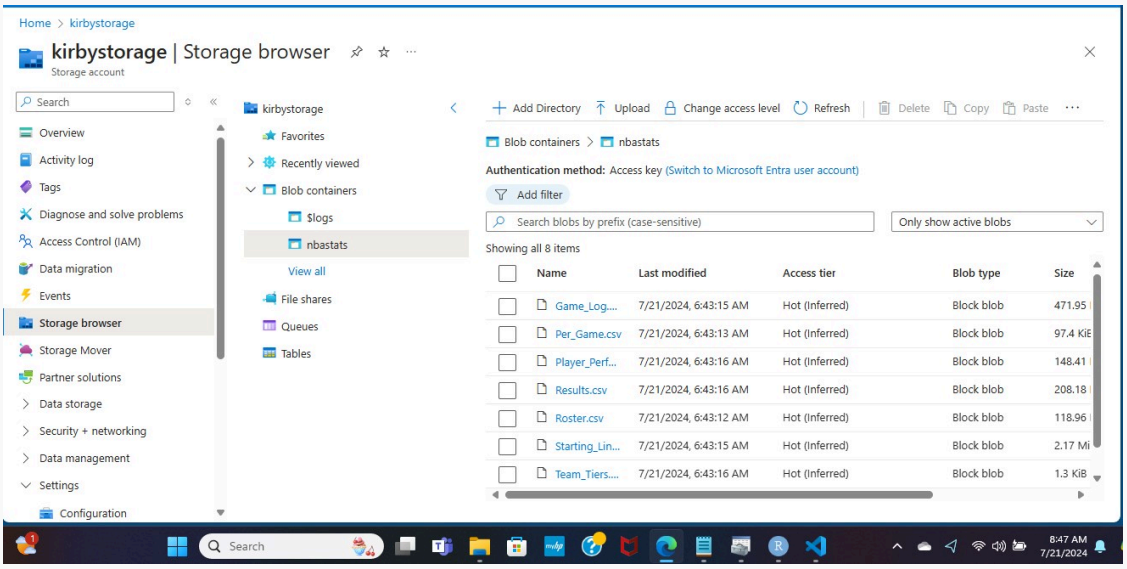
Microsoft Azure Storage before upload

```
account_name <- Sys.getenv("AZURE_STORAGE_ACCOUNT_NAME")
account_key <- Sys.getenv("AZURE_STORAGE_ACCOUNT_KEY")
blob_service <- Sys.getenv("AZURE_BLOB_SERvICE")


blob_endpoint <- storage_endpoint(
  sprintf(blob_service, account_name),
  key=account_key
)
```

```
## Warning in sprintf(blob_service, account_name): one argument not used by
## format 'https://kirbystorage.blob.core.windows.net/'
container_name <- "nbastats"

blob_container <- blob_container(blob_endpoint, container_name)
```

```


data_frames <- list(

  Roster = NBA_Roster,

  Per_Game = NBA_Per_Game,

  Totals = NBA_Totals,

  Starting_Lineups = NBA_Starting_Lineups,

  Game_Log =
  NBA_Game_Log,
  Results =
  NBA_Results,
  Team_Tiers =
  NBA_team_tiers,
  Player_Performance = Player_Performance_Long

)


temp_dir <- tempdir()

for (name in names(data_frames)) {
  file_path <- file.path(temp_dir, paste0(name,
  ".csv"))
  write.csv(data_frames[[name]], file_path, row.names =
  FALSE)
  upload_blob(blob_container, src=file_path, dest=paste0(name,
  ".csv"))
}

## |

|                                                                      | 0%   |
|=========                                                             | 13%  |
|==================                                                    | 27%  |
```

```
|============================                                          | 40%   |
|==================================                                    | 54%   |
|==========================================                            | 67%   |
|====================================================                  | 81%   |
|============================================================          | 94%   |
|======================================================================| 100%

##  |

|                                                                      | 0%    |
|===========                                                           | 16%   |
|======================                                                | 33%   |
|================================                                      | 49%   |
|========================================                              | 66%   |
|==================================================                    | 82%   |
|====================================================================  | 99%   |
|======================================================================| 100%
```



Microsoft Azure Storage after upload

Azure SQL Tables

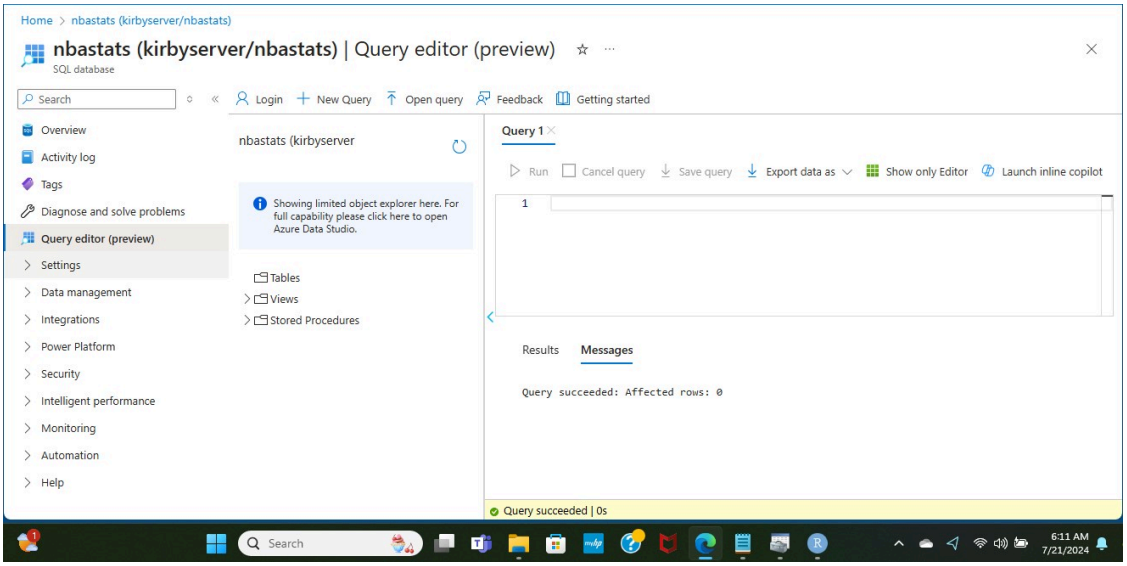
Retrieve connection details from environment variables.

Construct the connection string.

Connect to the database.

Check connection. (optional)

```
server                                  <-
Sys.getenv("AZURE_DB_SERVER")
database                                    <-
Sys.getenv("AZURE_DB_DATABASE")
user <- Sys.getenv("AZURE_DB_USER")
password <- Sys.getenv("AZURE_DB_PASSWORD")

connection_string <- sprintf(
  "Driver={ODBC Driver 17 for SQL
Server};Server=tcp:%s,1433;Database=%s;Uid=%s;Pwd=%s;Encrypt=yes;TrustServerC
ertificate=no;Connection Timeout=30;",
  server, database, user, password
)


azure_conn <- odbcDriverConnect(connection_string)


print(azure_conn)
## RODBC Connection 1
## Details:
## case=nochange
## DRIVER=ODBC Driver 17 for SQL Server
##   SERVER=tcp:kirbyserver.database.windows.net,1433
##   UID=kirbyserver
##   PWD=******
##   APP=RStudio
##   WSID=LAPTOP-Q08GOMEC
##   DATABASE=nbastats
##   Encrypt=yes
```

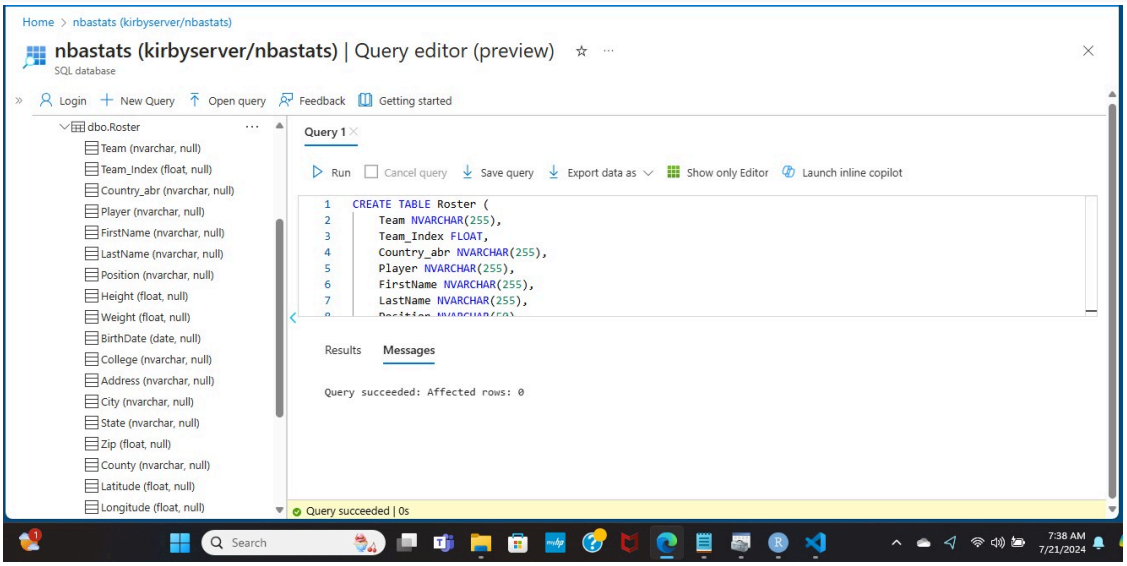


Azure SQL Tables before upload

There are a couple of options to uploading data into the Platforms SQL server. one way is to use SQL to create the tables and uplaod or you could use SQL syntax to create and insert the data into the tables. We can explore tthat more later…

For now, insert data into the pre-existing table using the `sqlSave` function.

```
sqlSave(azure_conn, NBA_Roster, tablename = "Roster", rownames = FALSE,
append = TRUE)
```



Azure SQL Roster Table Creation (In Server)

Create a function to convert column types to Azure SQL-compatible types. Sometimes server Sql platforms have slight variations in how they handle data types.

Convert data types.

```
ework_data_types <- function(df) {
  for (col in names(df)) {
    if (is.factor(df[[col]])) {
      df[[col]] <- as.character(df[[col]])
    } else if (is.integer(df[[col]])) {
      df[[col]] <- as.numeric(df[[col]])

    } else if (is.logical(df[[col]])) {

      df[[col]] <- as.numeric(df[[col]])

    } else if (inherits(df[[col]], "Date")) {

      df[[col]] <- as.character(df[[col]])
    } else if (inherits(df[[col]], "POSIXct") || inherits(df[[col]],
"POSIXt")) {
      df[[col]] <- as.character(df[[col]])
    }
  }

  return(df)
```

```
NBA_Starting_Lineups <- rework_data_types(NBA_Starting_Lineups)
NBA_Game_Log <- rework_data_types(NBA_Game_Log)

NBA_Results <- rework_data_types(NBA_Results)

NBA_team_tiers <- rework_data_types(NBA_team_tiers)
}
```

We could load individual data frames.

```
sqlSave(azure_conn, NBA_Per_Game, tablename = "Per_Game", rownames =
FALSE,
append = FALSE)



sqlSave(azure_conn, NBA_Totals, tablename = "Totals", rownames = FALSE,
append = FALSE)



sqlSave(azure_conn, NBA_Starting_Lineups, tablename = "Starting_Lineups",
rownames = FALSE, append = FALSE)


sqlSave(azure_conn, NBA_team_tiers, tablename = "team_tiers", rownames =
FALSE, append = FALSE)


sqlSave(azure_conn, Player_Performance_Long, tablename =
"Player_Performance", rownames = FALSE, append = FALSE)
```

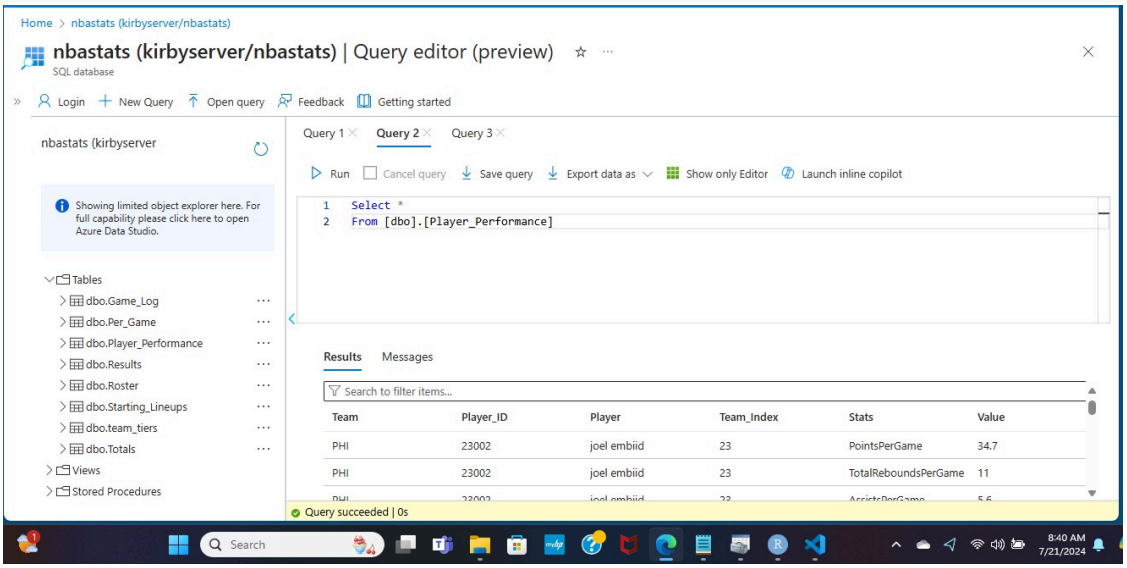
Or, we could list the data frames and their corresponding table names.

Load data frames into the function.

Then, load the list of data frames into the database.

```
data_frames       <-
list(
  NBA_Game_Log = "Game_Log",
  NBA_Results = "Results"
)
```

```r
load_to_azure <- function(df, table_name, conn) {
  sqlSave(conn, df, tablename = table_name, rownames = FALSE, append = TRUE)
}

for (df_name in names(data_frames)) {
  df <- get(df_name)
  table_name <- data_frames[[df_name]]
  load_to_azure(df, table_name, azure_conn)
}
```



Azure SQL Database Complete

Close the connection.

```r
close(azure_conn)
```

JSON & JavaScript

JSON File creation

Next we are going to set up some example files that we are going to need to complete a data transfers, through JSON and JavaScript.

First we are going to create our JSON File to handle our data types

Convert dataframe to JSON

Define a function to convert R data types to BigQuery types

Get schema from dataframe

Construct BigQuery schema JSON

Write schema to JSON file

```
json_data  <-  toJSON(NBA_Roster,  pretty  =
TRUE)


to_bigquery_type <- function(class_name) {
  switch(class_name,
         character = "STRING",

         numeric = "INTEGER",

         Date = "DATE",

         "UNKNOWN")
}


schema <- lapply(NBA_Roster, function(x) {
  s <- list()
  s$name <- names(x)
  s$type <- to_bigquery_type(class(x))
  s

}
)


json_schema <- jsonlite::toJSON(list(`BigQuery Schema` = schema), pretty =
TRUE
)


writeLines(json_schema, "NBA_roster_schema.json")
```

JavaScript File Creation

Next we create our JavaScript file to handle the columns and the rows

Open file in write mode

Write JavaScript UDF function

Close the file connection

```
con <- file("NBA_roster_schemaUDF.js", open = "w")


writeLines('function transform(line) {', con)
writeLines(' var values = line.split(\',\');', con)

writeLines(' var obj = new Object();', con)


for (field in schema) {
  writeLines(paste0(' obj.', field$name, ' = values[', field$name, ' -
1];'), con)




writeLines(' var jsonString = JSON.stringify(obj);', con)
writeLines(' return jsonString;', con)
```

```
writeLines('}', con)


close(con)
}
```

SQL Script Creation

INSERT using SQL

Create function to escape single quotes. Because most SQL servers only recognize Single Quotations for separations through schema.

Apply the escaping function to the character columns.

```
escape_quotes <- function(x) {
  gsub("'", "''", x)


NBA_Roster$Team <- sapply(NBA_Roster$Team, escape_quotes)
NBA_Roster$Country_abr <- sapply(NBA_Roster$Country_abr, escape_quotes)
NBA_Roster$Player <- sapply(NBA_Roster$Player, escape_quotes)

NBA_Roster$FirstName <- sapply(NBA_Roster$FirstName, escape_quotes)

NBA_Roster$LastName <- sapply(NBA_Roster$LastName, escape_quotes)

NBA_Roster$Position <- sapply(NBA_Roster$Position, escape_quotes)

NBA_Roster$College <- sapply(NBA_Roster$College, escape_quotes)

NBA_Roster$Address <- sapply(NBA_Roster$Address, escape_quotes)

NBA_Roster$City <- sapply(NBA_Roster$City, escape_quotes)
NBA_Roster$State <- sapply(NBA_Roster$State, escape_quotes)
NBA_Roster$County <- sapply(NBA_Roster$County, escape_quotes)
```

```
NBA_Roster$Country <- sapply(NBA_Roster$Country, escape_quotes)

NBA_Roster$Country_abr3 <- sapply(NBA_Roster$Country_abr3,
escape_quotes)


}
```

Create the SQL INSERT statement, containing all the data ready for copy and paste into the query.

Save to a text file, for storage and to make it easier to copy and paste.

```
sql_values <- apply(NBA_Roster, 1, function(row) {
  sprintf(

    "('%s', %d, '%s', '%s', '%s', '%s', '%s', %d, %d, '%s', '%s', '%s', '%s',
'%s', %d, '%s', %.13f, %.13f, '%s', '%s', %d, %.1f, %.1f, %d)",

    row["Team"], as.integer(row["Team_Index"]), row["Country_abr"],
row["Player"], row["FirstName"], row["LastName"], row["Position"],
    as.integer(row["Height"]), as.integer(row["Weight"]),
as.character(row["BirthDate"]), row["College"], row["Address"], row["City"],
row["State"], as.integer(row["Zip"]),
    row["County"], as.numeric(row["Latitude"]), as.numeric(row["Longitude"]),
row["Country"], row["Country_abr3"], as.integer(row["Country_num"]),

    as.numeric(row["Cty_Latitude"]), as.numeric(row["Cty_Longitude"]),

as.integer(row["Player_ID"])
  )
})
```

```
sql_query <- paste(
  "INSERT INTO Roster (Team, Team_Index, Country_abr, Player, FirstName,
LastName, Position, Height, Weight, BirthDate, College, Address, City,
State,
Zip,    County,    Latitude,    Longitude,    Country,    Country_abr3,
Country_num,
Cty_Latitude, Cty_Longitude, Player_ID) VALUES",

  paste(sq
  l_values
  ,
  collapse
  = ",\n")


write(sql_query, file = "insert_roster.sql")

)
```

Conclusion

ETL Overview

In this overview of the Transform and Load 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 Load the Data.

Portfolio Project

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