Assignment 3 - Baseball Database Analysis

Yifu Wu (916619585) *

11/19/2020

In this assignment, I will explore the baseball database using both SQL and R. This database contains thousands of pitching, hitting, and fielding statistics for Major League Baseball from 1871 through 2013, which was created by Sean Lahman. The documentations of the database can be found at http://www.seanlahman.com/files/database/readme2013.txt. In this report, I will answer 20 question using SQL, and then verify the results with R.
Note: We send the SQL queries from R to the database via ***dbGetQuery()*** *which will not be shown in the following questions.*

1 Question 1

Question: ***What years does the data cover? Are there data for each of these years?***
The question asks to find the years that the data cover. However, the database contains many tables, we select one of the main table, which is the Batting table to see how many years do does the table contain and also find out whether there exists data for each year. To do this, we need to first find the **DISTINCT** yearID in the table since we have several observations in each year. Other than **COUNT** the number of **DISTINCT** yearID, we also list all years included in the table.
First count the number of distinct year:

```
SELECT COUNT(DISTINCT yearID)
FROM Batting
```

Then list all years included in the table:

```
SELECT DISTINCT yearID
FROM Batting
ORDER BY yearID
```

The output of SQL shows that there are totally 143 years covered in the table ranging from 1871 to 2013. Then we want to find out whether other tables also have the same range of data. We choose two important tables which are Managers and Salaries table. We **COUNT** the **DISTINCT** yearID and also compute the **MAX** and **MIN** yearID in each tables to get the year range.
First explore the Managers table:

```
SELECT COUNT(DISTINCT yearID), MAX(DISTINCT yearID), MIN(DISTINCT yearID)
FROM Managers
```

Then the Salaries table:

*Department of Statistics, UC Davis and ifuwu@ucdavis.edu

```
SELECT COUNT(DISTINCT yearID), MAX(DISTINCT yearID), MIN(DISTINCT yearID)
FROM Salaries
```

From the output of SQL we know that the Managers table also has 143 years' data ranging from 1871 to 2013, while the Salaries table only contains 29 years' data ranging from 1985 to 2013.
To do this in R, We get the minimal and maximal year in Batting, Managers and Salaries table via **min()** and **max()** fucntion in R, then using **unique()** and **length()** function to compute the number of distinct years.

```
Batting <- dbGetQuery(baseball, "SELECT * FROM Batting")
c(From = min(Batting$yearID),
  To = max(Batting$yearID),
  Duration = length(unique(Batting$yearID)))
Managers <- dbGetQuery(baseball, "SELECT * FROM Managers")
c(From = min(Managers$yearID),
  To = max(Managers$yearID),
  Duration = length(unique(Managers$yearID)))
Salaries <- dbGetQuery(baseball, "SELECT * FROM Salaries")
c(From = min(Salaries$yearID),
  To = max(Salaries$yearID),
  Duration = length(unique(Salaries$yearID)))
```

2 Question 2

Question: ***How many (unique) people are included in the database? How many are players, managers, etc?***
This question asks to find the number of unique people included in the database, and also find number of players and managers. To answer the question we also need to use **COUNT DISTINCT** to calculate the number of unique people. First we select all tables in the database which contain the playerID field, it can give us all unique people in the dataset. We will use **UNION** to combine all tables together.

```
SELECT COUNT(DISTINCT [unique].playerID)
FROM(
    SELECT PlayerID FROM Master
    UNION SELECT PlayerID FROM Batting
    UNION SELECT PlayerID FROM Pitching
    UNION SELECT PlayerID FROM Fielding
    UNION SELECT PlayerID FROM AllstarFull
    UNION SELECT PlayerID FROM HallOfFame
    UNION SELECT PlayerID FROM Managers
    UNION SELECT PlayerID FROM BattingPost
    UNION SELECT PlayerID FROM PitchingPost
    UNION SELECT PlayerID FROM FieldingOF
    UNION SELECT PlayerID FROM ManagersHalf
    UNION SELECT PlayerID FROM Salaries
    UNION SELECT PlayerID FROM AwardsManagers
    UNION SELECT PlayerID FROM AwardsPlayers
    UNION SELECT PlayerID FROM AwardsShareManagers
```

```
    UNION SELECT PlayerID FROM AwardsSharePlayers
    UNION SELECT PlayerID FROM FieldingPost
    UNION SELECT PlayerID FROM Appearances
    UNION SELECT PlayerID FROM SchoolsPlayers
) AS [unique]")
```

The result shows that there are totally 18359 unique people in the database.
Then we will count the total number of managers in the database. Different from the previous step, we only choose 4 tables that contain managers' information. We will also use **COUNT DISTINCT** to compute the number of unique managers and use **UNION** to combine these 4 tables. For counting the number of unique manager or player, we do not use HallofFame table since this contains people other than manager and player, such as the Umpire.

```
SELECT COUNT(DISTINCT manager.playerID)
FROM(
    SELECT playerID FROM Managers
    UNION SELECT playerID FROM ManagersHalf
    UNION SELECT playerID FROM AwardsManagers
    UNION SELECT PlayerID FROM AwardsShareManagers
) AS manager
```

The result shows that there are totally 682 unique managers in the database.
Then we will count the total number of players in the database. Different from the previous step, we choose 13 tables that contain players' information. We will also use **COUNT DISTINCT** to compute the number of unique players and use **UNION** to combine these 13 tables.

```
SELECT COUNT(DISTINCT player.playerID)
FROM(
    SELECT PlayerID FROM Batting
    UNION SELECT PlayerID FROM Pitching
    UNION SELECT PlayerID FROM Fielding
    UNION SELECT PlayerID FROM AllstarFull
    UNION SELECT PlayerID FROM BattingPost
    UNION SELECT PlayerID FROM PitchingPost
    UNION SELECT PlayerID FROM FieldingOF
    UNION SELECT PlayerID FROM Salaries
    UNION SELECT PlayerID FROM AwardsPlayers
    UNION SELECT PlayerID FROM AwardsSharePlayers
    UNION SELECT PlayerID FROM FieldingPost
    UNION SELECT PlayerID FROM Appearances
    UNION SELECT PlayerID FROM SchoolsPlayers
) AS player
```

The result shows that there are totally 18137 unique players in the database.
To do this in R, we just need to select columns from each table and combine them together. Then using**unique()** and **length()** functions to compute the number of distinct people.

```
all.ID <- c(Master$playerID, Batting$playerID, Pitching$playerID,
            Fielding$playerID, AllstarFull$playerID, HallOfFame$playerID,
            Managers$playerID, BattingPost$playerID, PitchingPost$playerID,
```

```
            FieldingOF$playerID, ManagersHalf$playerID, Salaries$playerID,
            AwardsManagers$playerID, AwardsPlayers$playerID, AwardsShareManagers$playerID,
            AwardsSharePlayers$playerID, FieldingPost$playerID, Appearances$playerID,
            SchoolsPlayers$playerID)
length(unique(all.ID))
manager.ID <- c(Managers$playerID, ManagersHalf$playerID, AwardsManagers$playerID,
                AwardsShareManagers$playerID)
length(unique(manager.ID))
player.ID <- c(Batting$playerID, Pitching$playerID, Fielding$playerID,
               AllstarFull$playerID, BattingPost$playerID, PitchingPost$playerID,
               FieldingOF$playerID, Salaries$playerID, AwardsPlayers$playerID,
               AwardsSharePlayers$playerID, FieldingPost$playerID, Appearances$playerID,
               SchoolsPlayers$playerID)
length(unique(player.ID))
```

3 Question 3

Question: ***How many players became managers?***
This question asks to count the number of players that become the manager. We can have two different explanations for this question. One is that we want to know the number of players who become the player-manager. In the Managers table, the plyrMgr field indicates that whether the player is a player-manager, which means that a member of a baseball team who simultaneously holds both playing and managing duties. We can calculate the number of unique players who is the player-manager by using **COUNT DISTINCT** and **GROUP BY** the playMgr field. The number of "Y" in the result is what we want. Although some players are the player-manager in some years, and not in other years, we regard them as a player-manager in this question.

```
SELECT COUNT(DISTINCT playerID) AS Count, plyrMgr AS Type
FROM Managers
GROUP BY plyrMgr
```

From the output we know that there are 247 "Y" and 515 "N" in the table, which means that there are 247 players are player-manager.
The other explanation of the question is that we count both the number of player-manager and the number of players who later become a manager. To avoid over counting, we use the debut field in the Master table which indicates the date that player made first major league appearance. In this field, if the value is 0, we can say that this people is not a player. Based on that, we **LEFT JOIN** the the Masters table to the Managers table (all observations in this table are manager) by the playerID, and select observation **WHERE** the debut is not equal to 0. Then we use **COUNT DISTINCT** to count the number of unique managers.

```
SELECT COUNT(DISTINCT playerID) AS count
FROM(
    SELECT *
    FROM Managers
    LEFT JOIN Master ON Managers.playerID = Master.playerID
)
WHERE debut != 0
```

Then we verify it in R. To verify the first explanation, we need to select the observations which value in plyMgr field is "Y", then use **unique()** and **length()** functions to compute the number of distinct managers. For the second explanations we need to use **merge()** function to merge Managers and Master table into one, then select the observations which value in debut field is not 0. Then using **unique()** and **length()** functions to compute the number of distinct managers.

```
Managers <- dbGetQuery(baseball, "SELECT * FROM Managers")
length(unique(Managers$playerID))
plyrMgr_yes <- Managers[which(Managers$plyrMgr == 'Y'), ]
length(unique(plyrMgr_yes$playerID))
plyrMgr_no <- Managers[which(Managers$plyrMgr == 'N'), ]
length(unique(plyrMgr_no$playerID))
total_number_manager <- merge(Managers,Master,by.x="playerID",by.y="playerID",all.x=TRUE)
total_number_manager <- total_number_manager[which(total_number_manager$debut != 0), ]
length(unique(total_number_manager$playerID))
```

4 Question 4

Question: ***What team won the World Series in 2010? Include the name of the team, the league and division.***
This question asks to list the information of the team that win the World Series in 2010. We can simply use the Teams table and select the the yearID field which value is 2010 and WSWin field which value is "Y" by using the **WHERE** operation.

```
SELECT yearID AS Year, teamID AS Team, name AS Name, lgID AS League, divID AS Division
FROM Teams
WHERE yearID = '2010' AND WSWin = 'Y'
```

The output is shown in the table below:

Year	Team ID	Team Name	League	Division
2010	SFN	San Francisco Giants	NL	W

To do this in R, we just need to select the team that the yearID equals to 2010, and the WSWin equals to "Y". Then select the columns we need (yearID, teamID, name, lgID, divID).

```
Teams <- dbGetQuery(baseball, "SELECT * FROM Teams")
WS2010_win <- Teams[which(Teams$yearID == 2010 & Teams$WSWin == "Y"), ]
WS2010_win[1, c("yearID", "teamID", "name", "lgID", "divID")]
```

5 Question 5

Question: ***Compute the table of World Series winners for all years, again with the name of the team, league and division.***
In this question, we need to find winners in the World Series for all years in the database. From the website we know that from 1969, the leagues begin to split into two divisions (East and West), so the Division field will be NA for those winners before 1969. Since there are some missing data in the Teams table, we need to **LEFT JOIN** the SeriesPost table to get the all winners in the

database. We join two table **ON** yearID, and find the observations which round field is "WS" by using **WHERE** operation. After doing this, we can **SELECT** corresponding fields which are **GROUP BY** yearID. We also need to make sure the teamID correct matches the winner's teamID since it may be different in two tables. We will use **WHERE** to control that we match the correct teamID.

```
SELECT yearID AS Year, teamID AS TeamWin, name AS NameWin,
       lgIDwinner AS LeagueWin, divID AS DivisionWin
FROM(
    SELECT * FROM Teams
    LEFT JOIN SeriesPost
    ON Teams.yearID = SeriesPost.yearID
    WHERE round = 'WS'
)
WHERE WSWin = 'Y' OR teamID = teamIDwinner OR franchID = teamIDwinner
      OR teamIDBR = teamIDwinner OR teamIDlahman45 = teamIDwinner
      OR teamIDretro = teamIDwinner
GROUP BY yearID
```

Since we do not have enough space to show all 116 winners from the database, we randomly choose to show 5 of them as examples.

Year	Team ID	Team Name	League	Division
1884	PRO	Providence Grays	NL	NA
1906	CHA	Chicago White Sox	AL	NA
1969	NYN	New York Mets	NL	E
1990	CIN	Cincinnati Reds	NL	W
2013	BOS	Boston Red Sox	AL	E

To do this in R, we need to first merge two tables by yearID using **merge()** function. Then we select observations based on the conditions shown above with **which()** function. The last step is to select useful columns to show the results.

```
SeriesPost <- dbGetQuery(baseball, "SELECT * FROM SeriesPost")
Teams <- dbGetQuery(baseball, "SELECT * FROM Teams")
series_ws <- SeriesPost[which(SeriesPost$round == "WS"), ]
merge_team_series <- merge(Teams,series_ws,by.x="yearID",by.y="yearID",all.X=TRUE)
WS_win<-merge_team_series[which(merge_team_series$WSWin=="Y" |
                          merge_team_series$teamID==merge_team_series$teamIDwinner |
                          merge_team_series$franchID==merge_team_series$teamIDwinner |
                          merge_team_series$teamIDBR==merge_team_series$teamIDwinner |
                          merge_team_series$teamIDlahman45==merge_team_series$teamIDwinner |
                          merge_team_series$teamIDretro==merge_team_series$teamIDwinner), ]
WS_win[, c("yearID", "teamID", "name", "lgID", "divID")]
```

6 Question 6

Question: ***What team lost the World Series each year? Again, include the name of the team, league and division.***

Similar with the previous question, we need to find the teams which lose the World Series each year. We first still need to combine the Teams and SeriesPost table using **LEFT JOIN** function **ON** the yearID. We also choose observations which round field is "WS" by using **WHERE** operation. Then instead of **SELECT** fields based on the winner's teamID, we need to make sure that the teamID matches the loser's teamID, which is the only different part from question 5. To get the loser for each year, we also need to use **GROUP BY** operation on yearID.

```
SELECT yearID AS Year, teamIDloser AS TeamLose, name AS NameLose,
       lgIDloser AS LeagueLose, divID AS DivisionLose
FROM(
    SELECT * FROM Teams
    LEFT JOIN SeriesPost
    ON Teams.yearID = SeriesPost.yearID
    WHERE round = 'WS'
)
WHERE teamID = teamIDloser OR franchID = teamIDloser OR teamIDBR = teamIDloser
      OR teamIDlahman45 = teamIDloser OR teamIDretro = teamIDloser
GROUP BY yearID
```

Since we do not have enough space to show all 116 winners from the database, we will choose 5 year records as examples. In this question, I will choose the same year as question 5.

Year	Team ID	Team Name	League	Division
1884	NYP	New York Metropolitans	AA	NA
1906	CHN	Chicago Cubs	NL	NA
1969	BAL	Baltimore Orioles	AL	E
1990	OAK	Oakland Athletics	AL	W
2013	SLN	St. Louis Cardinals	NL	C

We will use the same approach described in question 5 to verify the results in R.

```
SeriesPost <- dbGetQuery(baseball, "SELECT * FROM SeriesPost")
Teams <- dbGetQuery(baseball, "SELECT * FROM Teams")
series_ws <- SeriesPost[which(SeriesPost$round == "WS"), ]
merge_team_series <- merge(Teams,series_ws,by.x="yearID",by.y="yearID",all.X=TRUE)
WS_lose<-merge_team_series[which(merge_team_series$teamID==merge_team_series$teamIDloser |
                           merge_team_series$franchID==merge_team_series$teamIDloser |
                           merge_team_series$teamIDBR==merge_team_series$teamIDloser |
                           merge_team_series$teamIDlahman45==merge_team_series$teamIDloser |
                           merge_team_series$teamIDretro==merge_team_series$teamIDloser), ]
WS_lose[, c("yearID", "teamIDloser", "name", "lgID", "divID")]
```

7 Question 7

Question: ***Do you see a relationship between the number of games won in a season and winning the World Series?***
In this question, the SQL code is rather similar with question 5. The only different point is that

we do not need to **SELECT** the teamID, league, etc., we just need to choose the W field in the Teams table which indicates the number of wins in a year (season).

```
SELECT W AS numberWin
FROM(
    SELECT * FROM Teams
    LEFT JOIN SeriesPost
    ON Teams.yearID = SeriesPost.yearID
    WHERE round = 'WS'
)
WHERE WSWin = 'Y' OR teamID = teamIDwinner OR franchID = teamIDwinner
      OR teamIDBR = teamIDwinner OR teamIDlahman45 = teamIDwinner
      OR teamIDretro = teamIDwinner
GROUP BY yearID
```

To see the relationship between the number of games won in a season and the winning the World Series, I use histogram to explore the distribution of the number of wins.

```
hist(relationship$numberWin, xlab = "Number of Win",
     main = "Number of Win in a season for the Champion of the World Series")
```

The plot is shown below:

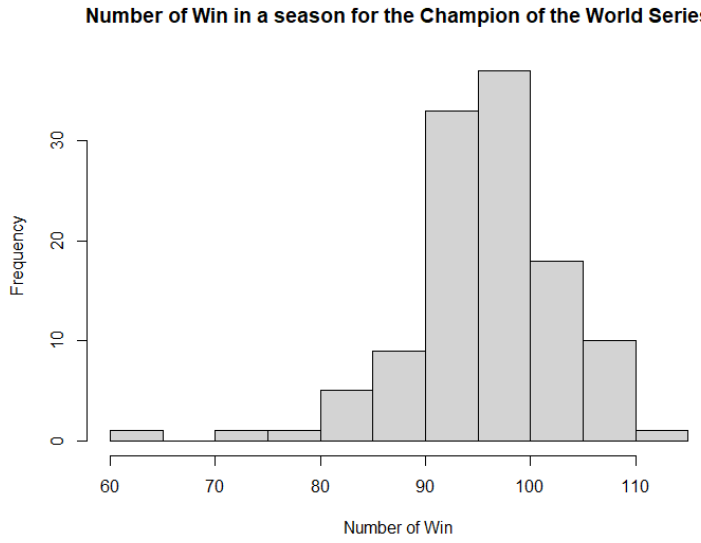

Figure 1: Number of Win in a season for the Champion of the World Series

From the plot we know that for most of the champion of the World series, the total number of wins of a season is around 90 to 100. Although the total number of games in a season is different among years, we can still speculate that the more games a team win, the more likely to be a champion.
The R code is also similar with that in question 5. We need to use **hist()** to draw the histogram.

```
SeriesPost <- dbGetQuery(baseball, "SELECT * FROM SeriesPost")
Teams <- dbGetQuery(baseball, "SELECT * FROM Teams")
```

```
series_ws <- SeriesPost[which(SeriesPost$round == "WS"), ]
merge_team_series <- merge(Teams,series_ws,by.x="yearID",by.y="yearID",all.X=TRUE)
WS_win<-merge_team_series[which(merge_team_series$WSWin=="Y" |
                          merge_team_series$teamID==merge_team_series$teamIDwinner |
                          merge_team_series$franchID==merge_team_series$teamIDwinner |
                          merge_team_series$teamIDBR==merge_team_series$teamIDwinner |
                          merge_team_series$teamIDlahman45==merge_team_series$teamIDwinner |
                          merge_team_series$teamIDretro==merge_team_series$teamIDwinner), ]
hist(WS_win$W, xlab = "Number of Win",
     main = "Number of Win in a season for the Champion of the World Series")
```

8 Question 8

Question: ***In 2003, what were the three highest salaries? (We refer here to unique salaries, i.e., there may be several players getting the exact same amount.) Find the players who got any of these 3 salaries with all of their details?***
In this question we need to first find out the top 3 highest salaries in 2003. Since we need to find the unique salaries, so we have to use **SELECT DISTINCT** operation in the salary field. Also we need to use **WHERE** operation to point out the yearID we need. Since we need to find out the top 3 highest salaries, we need to get the descending order of our result by using **ORDER BY** operation with **DESC**. Also using **LIMIT** operation to make sure we select the top 3 salaries.

```
SELECT DISTINCT salary
FROM Salaries
WHERE yearID = '2003'
ORDER BY salary DESC LIMIT 3
```

The results show that the top 3 salaries in 2003 were 22000000, 20000000 and 18700000. The next step is to find the players who get these salaries. Since we need to provide some useful information about the players, we **SELECT** 16 fields including name, birth country, team, number of wins in 2003, etc. To do this we need to **LEFT JOIN** 3 tables which are Master, Appearances and Teams to the Salaries table. Also we need to rank them in order of descending salary as previous step.

```
SELECT Salaries.yearID, Salaries.playerID, Master.nameFirst, Master.nameLast,
       Master.birthYear, Master.birthCountry, Master.weight, Master.height,
       Salaries.teamID, Teams.name, Salaries.lgID, Salaries.salary,
       Appearances.G_all, Teams.W, Teams.WSWin, Teams.LgWin
FROM Salaries
LEFT JOIN Master ON Salaries.playerID = Master.playerID
LEFT JOIN Appearances ON Master.playerID = Appearances.playerID
          AND Salaries.yearID = Appearances.yearID
LEFT JOIN Teams ON Teams.teamID = Appearances.teamID
          AND Salaries.yearID = Teams.yearID
WHERE Salaries.yearId = '2003' AND (Salaries.salary = '22000000'
      OR Salaries.salary = '20000000' OR Salaries.salary = '18700000')
ORDER BY Salaries.salary DESC
```

The result table is shown below:

Year	2003	2003	2003
PlayerID	rodrial01	ramirma02	delgaca01
First Name	Alex	Manny	Carlos
Last Name	Rodriguez	Ramirez	Delgado
Birth Year	1975	1972	1972
Birth Country	USA	D.R.	P.R.
Weight	225	225	215
Height	75	72	75
TeamID	TEX	BOS	TOR
Team Name	Texas Rangers	Boston Red Sox	Toronto Blue Jays
League	AL	AL	AL
Salary	22000000	20000000	18700000
Number of Game	161	154	161
Number of Wins	71	95	86
World Series	N	N	N
League Championship	N	N	N

To do this in R, we need to use **distinct()** function in the **dplyr** package to identify the top 3 unique salaries, and using **order()** function to get the decreasing order. Then we need to use **merge()** function to combine 4 tables and select corresponding fields we need.

```
Salaries <- dbGetQuery(baseball, "SELECT * FROM Salaries")
Master <- dbGetQuery(baseball, "SELECT * FROM Master")
Appearances <- dbGetQuery(baseball, "SELECT * FROM Appearances")
Teams <- dbGetQuery(baseball, "SELECT * FROM Teams")
salary_2003 <- Salaries[which(Salaries$yearID == '2003'), ]
unique_salary_2003 <- dplyr::distinct(salary_2003, salary)
unique_salary_2003[order(unique_salary_2003, decreasing = TRUE)[1:3], ]
master_sub <- Master[, c("playerID", "birthYear", "birthCountry",
                         "nameFirst", "nameLast", "weight", "height")]
merge1 <- merge(salary_2003, master_sub, by.x = "playerID", by.y = "playerID", all.x = TRUE)
appearances_sub <- Appearances[which(Appearances$yearID == "2003"), c("playerID", "G_all")]
merge2 <- merge(merge1, appearances_sub, by.x = "playerID", by.y = "playerID", all.x = TRUE)
team_sub <- Teams[which(Teams$yearID == "2003"), c("teamID", "name", "W", "WSWin", "LgWin")]
player_information <- merge(merge2, team_sub, by.x = "teamID", by.y = "teamID", all.x = TRUE)
player_information[order(player_information$salary, decreasing = TRUE)[1: 3], ]
```

9 Question 9

Question: ***For 2010, compute the total payroll of each of the different teams. Next compute the team payrolls for all years in the database for which we have salary information. Display these in a plot.***

In the first part of the question, we need to calculate the total salary of each team in 2010. To do this we use **SELECT SUM** operation to compute the sum of the salary. We also need both **WHERE** and **GROUP BY** operations to make sure we select 2010 and get total salary for each teams.

```
SELECT teamID AS team, SUM(salary) AS total_payroll
FROM Salaries
WHERE yearID = '2010'
GROUP BY teamID
```

Then we use scatter plot to show the result we get from SQL:

```
ggplot(payroll_2010_sql, aes(x = team, y = total_payroll)) +
  geom_point() +
  coord_flip()
```

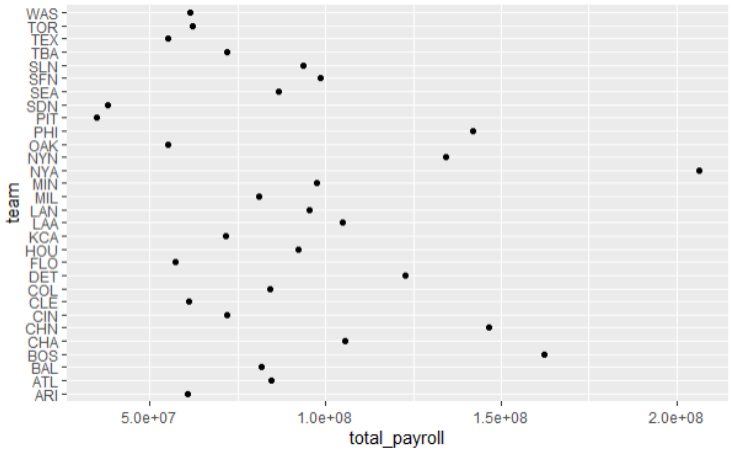

Figure 2: Total Payroll of All Teams in 2010

From the plot we know that the NYA has the highest total payroll which is more than 200000000 while the PIT has the lowest which is lower than 50000000. For most teams, the total payroll is between 50000000 and 100000000.

In the second part, we need to show the team total payroll for all years. The only different from the previous part is that we do not need to specify the yearID equals to 2010.

```
SELECT yearID AS year, teamID AS team, SUM(salary) AS total_payroll
FROM Salaries
GROUP BY teamID, yearID
```

We also use scatter plot to show the result:

```
ggplot(payroll_all_sql, aes(x = year, y = total_payroll, color = team)) +
  geom_point() +
  guides(col = guide_legend(nrow = 12))
```

From the scatter plot we know that as time goes on, the difference of total payroll between teams becomes bigger. We can also say that the overall salary becomes higher as time passed.

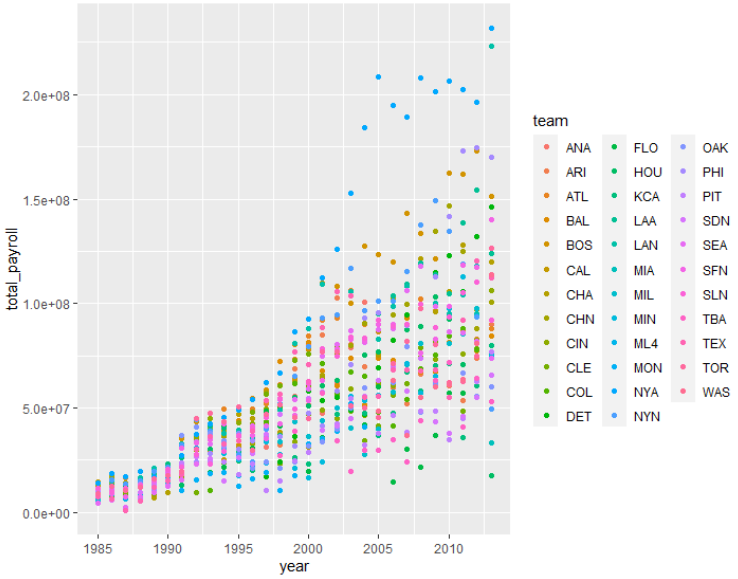

Figure 3: Total Team Payrolls for All Years

It is easy to verify the result in R with **dplry** package. Using **group_by()** function allow us to aggregate the data by teamID or yearID. Also we will use **summarise()** function with **sum()** function to calculate the total payroll in each groups. The approach for both parts in this question is rather similar.

```
Salaries <- dbGetQuery(baseball, "SELECT * FROM Salaries")
payroll_2010_r <- Salaries[which(Salaries$yearID == "2010"), ]
payroll_2010_r <- payroll_2010_r %>%
  group_by(teamID) %>%
  summarise(salary = sum(salary))
ggplot(payroll_2010_r, aes(x = teamID, y = salary)) +
  geom_point() +
  coord_flip()
payroll_all_r <- Salaries %>%
  group_by(teamID, yearID) %>%
  summarise(salary = sum(salary))
ggplot(payroll_all_r, aes(x = yearID, y = salary, color = teamID)) +
  geom_point() +
  guides(col = guide_legend(nrow = 12))
```

10 Question 10

Question: ***Which player has hit the most home runs? Show the number per year.***
In this question, we need to find the player that hit the most home runs in each year. This question is tricky since it is possible that several players hit the same number of home runs, so we need to list all of them. It appears to be a "greatest N per group" problem. We need to first get the maximum home runs per year using *MAX* and **GROUP BY** operations, then using them as a derived table, join them back by **INNER JOIN** operation to the source to get the observations matching the maximums. Since we want to know the name of the player, we also use **LEFT JOIN**

operation to combine the Batting and Master tables.

```
SELECT B.playerID, Master.nameFirst, Master.nameLast, B.yearID, Y.max_HR
FROM Batting AS B
INNER JOIN(
          SELECT playerID, yearID, MAX(HR) AS max_HR
          FROM Batting
          GROUP BY yearID
) AS Y ON B.yearID = Y.yearID AND B.HR = Y.max_HR
LEFT JOIN Master ON B.playerID = Master.playerID
ORDER BY B.yearID
```

Since we do not have much space to show all results, we randomly choose 3 years which are 1871, 1982 and 2013 to show the players that hit the most home runs in that year.

PlayerID	First Name	Last Name	Year	Number of Home Runs
meyerle01	Levi	Meyerle	1871	4
pikeli01	Lip	Pike	1871	4
treacfr01	Fred	Treacey	1871	4
jacksre01	Reggie	Jackson	1982	39
thomago01	Gorman	Thomas	1982	39
davisch02	Chris	Davis	2013	53

We can see from the result that in 1871 there were 3 players got the same highest home runs, and 2 players in 1982, and only 1 in 2013.

To do this in R, we need to first use **merge()** function to combine the Batting table with the Master table. Since the value in some cells are NA, which may cause miscalculation. So we first use **replace()** function to replace those NA with 0. Next we **select()** useful columns and group them by yearID with **group_by()** function. Then we try to filter the observations which value equal to the maximum value by using **filter()** function.

```
Batting <- dbGetQuery(baseball, "SELECT * FROM Batting")
Master <- dbGetQuery(baseball, "SELECT * FROM Master")
batting_with_name <- merge(Batting, Master[, c("playerID", "nameFirst", "nameLast")],
                           by.x = "playerID", by.y = "playerID", all.x = TRUE)
batting_with_name %>%
  replace(is.na(.), 0) %>%
  select(playerID, nameFirst, nameLast, yearID, HR) %>%
  group_by(yearID) %>%
  filter(HR == max(HR)) %>%
  arrange(yearID)
```

11 Question 11

Question: ***Has the distribution of home runs for players increased over the years?***
In this question, I want to show the total number of home runs in each year and explore whether it increases over the years. First we need to find the total home runs in each year using the **SUM**

and **GROUP BY** operations. Then I will draw a scatter plot with the smooth line to identify the trend over the years.

```
SELECT yearID as year, SUM(HR) AS total_HR
FROM Batting
GROUP BY yearID
```

Then we draw the scatter plot and the smooth line:

```
ggplot(HR_year_sql, aes(x = year, y = total_HR)) +
  geom_point() +
  geom_smooth()
```

The plot is shown below:

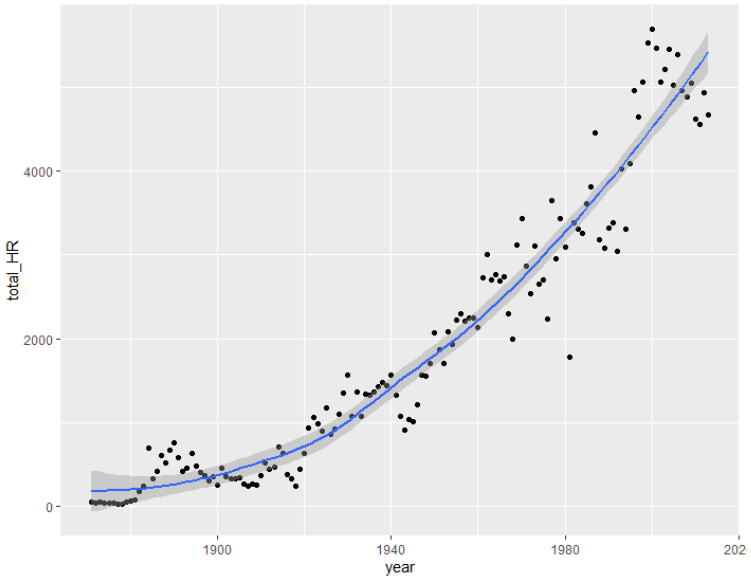

Figure 4: Distribution of Total Home Runs Over the Years

From the plot we know that the distribution of the total home runs increases over the years.

To do this in R, we first need to substitute NAs with 0 using **replace()** function. Then we can use **group_by()** and **summarise()** functions to compute the total home runs in each year.

```
Batting <- dbGetQuery(baseball, "SELECT * FROM Batting")
HR_year_r <- Batting %>%
  replace(is.na(.), 0) %>%
  group_by(yearID) %>%
  summarise(HR = sum(HR))
ggplot(HR_year_r, aes(x = yearID, y = HR)) +
  geom_point() +
  geom_smooth()
```

12 Question 12

Question: ***Are certain baseball parks better for hitting home runs?***
In this question we want to know whether some baseball parks have more hit runs than others.

The first step is to find the total home runs in different baseball parks over the years, and find the top 10 parks from it. To do that we need to **SELECT SUM** of the home runs **GROUP BY** each baseball parks. Then as we do in previous questions, we need to using **ORDER BY, DESC, LIMIT** operations to find the top 10 parks.

```
SELECT park, SUM(HR) AS total_HR
FROM Teams
GROUP BY park
ORDER BY SUM(HR) DESC LIMIT 10
```

The results are shown in the following table:

Park	Total Number of Home Runs
Wrigley Field	11859
Fenway Park II	11618
Sportsman's Park IV	7362
Yankee Stadium I	7186
Dodger Stadium	7103
Tiger Stadium	6406
County Stadium	6388
Cleveland Stadium	6220
Comiskey Park	6155
Shea Stadium	5891

The next step is to calculate the total number of home runs and find the top 5 parks in each year. Next we count the number of appearances of each park based on the result we just got. Then we will choose the top 5 parks which appear the most. To do this we need to first specifies a temporary named result set, known as a common table expression (CTE) using **WITH** operation. Next we will use the ranking operation which is **ROW_NUMBER()** to rank the home runs **PARTITION BY** the yearID. Then we can easily select top 5 by using **WHERE** operation.

```
WITH Top_Five AS(
                SELECT yearID AS year, park, HR, ROW_NUMBER()
                OVER(
                    PARTITION BY yearID
                    ORDER BY HR DESC
                ) AS Row_Number
                FROM Teams
)
SELECT year, park, HR
FROM Top_Five
WHERE Row_Number <= 5
```

Then we use the **table()** function in R to count the number of occurrence for each park.

```
table(top_five_parks_sql$park)[order(table(top_five_parks_sql$park),decreasing=TRUE)[1:5]]
```

The results are shown in the following table:

Park	Total Number of Occurrence
Polo Grounds IV	44
Yankee Stadium I	37
Fenway Park II	33
Baker Bowl	29
Sportsman's Park IV	23

Based on two tables we know that the Yankee Stadium I, Fenway Park II and Sportsman's Park IV are the top 10 parks which have the most number of home runs over the years. They are also the top 5 parks that appear most in the annual ranking. Therefore, we can say that these 3 parks are much better for hitting home runs.

To verify the first step in R, similar with previous, we use functions in **dplyr** package. Other than the functions we used before, we use **arrange()** function to order the number of home runs and use **top_n()** to select the top 10 parks. To verify the second step, we will use **slice()** function to select the top 5 parks that have most home runs in each year.

```
Teams <- dbGetQuery(baseball, "SELECT * FROM Teams")
Teams %>%
  group_by(park) %>%
  summarise(HR = sum(HR)) %>%
  arrange(desc(HR)) %>%
  top_n(10)
top_five_parks_r <- Teams %>%
                      select(yearID, park, HR) %>%
                      arrange(desc(HR)) %>%
                      group_by(yearID) %>%
                      slice(1: 5)
table(top_five_parks_r$park)[order(table(top_five_parks_r$park),decreasing=TRUE)[1:5]]
```

13 Question 13

Question: ***How many games do pitchers start in a season? Plot this against games finished in a season.***

In this question we will plot the number of games that pitchers start in a season against the number of games that pitchers finished in a season. To do this we need to use the Pitching table and use **SELECT SUM** operation to count the total number of game started and game finished by pitchers respectively. Since we need to know the number of games in a season, we need to use **GROUP BY** operation.

```
SELECT yearID as Season, SUM(GS) AS Game_Started, SUM(GF) AS Game_Finished
FROM Pitching
GROUP BY yearID
```

Then we find that there are NAs in the first 5 years of the game finished field, and to avoid error in the plot, we replace those with 0. Then we use **ggplot()** to plot the scatter plot between the number of game started and finished by pitchers.

```
pitcher_sf_sql[is.na(pitcher_sf_sql)] = 0
ggplot(pitcher_sf_sql, aes(x = Game_Started, y = Game_Finished, color = Season)) +
  geom_point()
```

The plot is shown below:

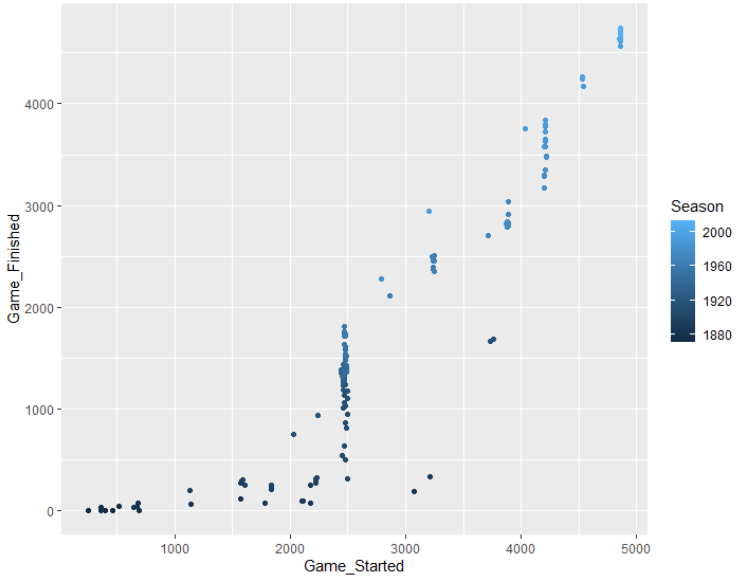

Figure 5: Number of Games Started Against Finished by Pitchers in a Season

From the plot we know that in early years, seldom games were finished by pitchers. As time goes on, more games are finished by pitchers. For recent years, the number of game started by pitchers does not have much difference with the number of game finished by pitcher.

To verify it in R, we also use functions in **dplyr** package discussed in the previous questions. Use **select()** to select useful fields, use **replace()** to replace NAs with 0, use **group_by()** and **summarise()** to compute the total number of games.

```
Pitching <- dbGetQuery(baseball, "SELECT * FROM Pitching")
pitcher_sf_r <- Pitching %>%
                  select(yearID, GS, GF) %>%
                  replace(is.na(.), 0) %>%
                  group_by(yearID) %>%
                  summarise(GS = sum(GS), GF = sum(GF))
ggplot(pitcher_sf_r, aes(x = GS, y = GF, color = yearID)) +
  geom_point()
```

14 Question 14

Question: ***How many games do pitchers win in a season?***
In this question we want to know the number of games pitchers win in each season. To do that in SQL, we need to use **SELECT SUM** operation to get the total number of wins, the **GROUP BY** operation can make sure that the we get the results for each season.

```
SELECT yearID AS Season, SUM(W) AS Number_Win
```

```
FROM Pitching
GROUP BY yearID
```

Then we use **ggplot()** to plot a scatter plot between year and number of wins in that year.

```
ggplot(pitcher_win_sql, aes(x = Season, y = Number_Win)) +
  geom_point()
```

The plot is shown below:

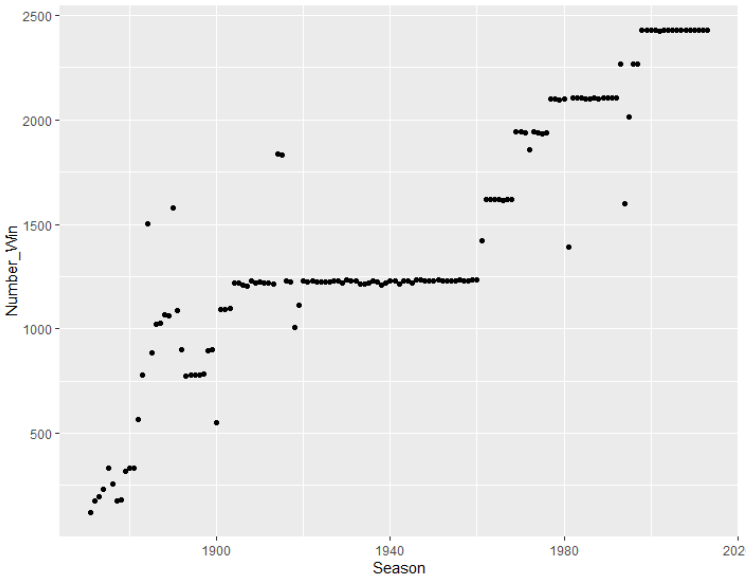

Figure 6: Number of Games Pitcher Win in each Season

From the plot we know that the total number of wins by pitchers increases over the years. Also from 1900 to 1960, the number of wins kept almost the same. Such situation also happened after 1960, for several consecutive years, the number of wins kept unchanged, but the time did not keep as long as it was in 1900.

To verify it in R, we still use the **dplyr** package, and the approach is rather similar with previous questions.

```
Pitching <- dbGetQuery(baseball, "SELECT * FROM Pitching")
pitcher_win_r <- Pitching %>%
                   select(yearID, W) %>%
                   replace(is.na(.), 0) %>%
                   group_by(yearID) %>%
                   summarise(W = sum(W))
ggplot(pitcher_win_r, aes(x = yearID, y = W)) +
  geom_point()
```

15 Question 15

Question: ***What are the top ten collegiate producers of major league baseball players? How many colleges are represented in the database?***

In this question we first need to find out the number of the colleges that included in the database. To do this one we only need to use **COUNT DISTINCT** operation to count the number of unique colleges in the Schools table.

```
SELECT COUNT(DISTINCT schoolID)
FROM Schools
```

The result shows that there are totally 749 colleges in the database.

The second part in this question is to find the top 10 collegiate producers of major league baseball (MLB) players. To do this we need to **LEFT JOIN** the SchoolPlayers table with the Schools table since the first table contains the information of players and the colleges they attended. We should use **GROUP BY** and **COUNT** operations to count the number of occurrence of each college in the table. Then use the **ORDER BY, DESC, LIMIT** operations to find the top 10 of them.

```
SELECT schoolName, COUNT(schoolName) AS Number_Players
FROM SchoolsPlayers
LEFT JOIN Schools ON SchoolsPlayers.schoolID = Schools.schoolID
GROUP BY SchoolsPlayers.schoolID
ORDER BY Number_Players DESC
LIMIT 10
```

The results are shown in the table below:

College Name	Number of Player Produced
University of Southern California	102
University of Texas at Austin	100
Arizona State University	95
Stanford University	82
University of Michigan	77
College of the Holy Cross	75
University of Notre Dame	70
University of Illinois at Urbana-Champaign	68
University of Arizona	66
University of California, Los Angeles	66

To verify the first part in R, we just need to use **length()** and **unique()** function to count the number of unique college name. For the second question, we use functions in **dplyr** package which are all explained in the previous questions.

```
Schools <- dbGetQuery(baseball, "SELECT * FROM Schools")
SchoolsPlayers <- dbGetQuery(baseball, "SELECT * FROM SchoolsPlayers")
length(unique(Schools$schoolID))
merge_school <- merge(SchoolsPlayers,Schools,by.x="schoolID",by.y="schoolID",all.x=TRUE)
merge_school %>%
  group_by(schoolName) %>%
  summarise(count = n()) %>%
  arrange(desc(count)) %>%
  top_n(10)
```

16 Question 16

Question: ***What players have pitched in the post season and also hit a home run in their career?***
In this question we will explore the PitchingPost to find the players that have pitched in the post season and also hit at least one home run in the career. To do this we need to combine three other tables which are Master, Batting and BattingPost with PitchingPost table using **LEFT JOIN** operation. If we want to know whether a pitcher hit a home run in the career, we need to make sure either the HR field in Batting or BattingPost table is larger than 0 by using the **WHERE** operation, since this two tables contain the information of home runs in games.

```
SELECT DISTINCT PitchingPost.playerID, Master.nameFirst, Master.nameLast
FROM PitchingPost
LEFT JOIN Master ON PitchingPost.playerID = Master.playerID
LEFT JOIN Batting ON PitchingPost.playerID = Batting.playerID
                     AND PitchingPost.yearID = Batting.yearID
                     AND PitchingPost.teamID = Batting.teamID
LEFT JOIN BattingPost ON PitchingPost.playerID = BattingPost.playerID
                         AND PitchingPost.yearID = BattingPost.yearID
                         AND PitchingPost.teamID = BattingPost.teamID
                         AND PitchingPost.round = BattingPost.round
WHERE Batting.HR > 0 OR BattingPost.HR > 0
```

The result table shows that there are 264 players hit a home run in their career if they have pitched in the post season. I will show 5 of them in the table below:

playerID	First Name	Last Name
aguilri01	Rick	Aguilera
aldrivi01	Vic	Aldridge
alexape01	Pete	Alexander
allenjo02	Johnny	Allen
anderbr02	Brian	Anderson

To verify it in R, we need to use **merge()** function to combine the 4 tables, then we get the result with duplicated playerID, so we just use **!duplicated()** function in R to get unique name of players

```
PitchingPost <- dbGetQuery(baseball, "SELECT * FROM PitchingPost")
Master <- dbGetQuery(baseball, "SELECT * FROM Master")
Batting <- dbGetQuery(baseball, "SELECT * FROM Batting")
BattingPost <- dbGetQuery(baseball, "SELECT * FROM BattingPost")
Master_sub <- Master[, c("playerID", "nameFirst", "nameLast")]
merge_1 <- merge(PitchingPost,Master_sub,by.x="playerID",by.y ="playerID",all.x=TRUE)
Batting_sub <- Batting[, c("playerID", "yearID", "teamID", "HR")]
merge_2 <- merge(merge_1, Batting_sub, by.x = c("playerID", "yearID", "teamID"),
                 by.y = c("playerID", "yearID", "teamID"), all.x = TRUE)
BattingPost_sub <- BattingPost[, c("playerID", "yearID", "teamID", "round", "HR")]
```

```
merge_3 <- merge(merge_2, BattingPost_sub,
                 by.x = c("playerID", "yearID", "teamID", "round"),
                 by.y = c("playerID", "yearID", "teamID", "round"), all.x = TRUE)
pitch_HR <- merge_3[which(merge_3$HR > 0 | merge_3$HR.y > 0),
                    c("playerID", "nameFirst", "nameLast")]
pitch_HR[!duplicated(pitch_HR$playerID), ]
```

17 Question 17

Question: ***How many players are there in each year, from 2000 to 2013? Do all teams have the same number of players?***

The first part in this question asks to count the number of players in each year. To do this we want to count the players that have detailed information on the positions they appeared at, so we choose the Appearances table. Like many other questions, we need to use **COUNT DISTINCT** and **GROUP BY** operations to count number of unique playerID in each year in the table. The **WHERE** operation can make sure we select the years between 2000 and 2013.

```
SELECT yearID AS year, COUNT(DISTINCT playerID) AS count
FROM Appearances
WHERE yearID >= '2000'
GROUP BY yearID
```

The results are shown in the table below:

Year	2000	2001	2002	2003	2004	2005	2006
Number of Players	1230	1220	1218	1230	1247	1237	1242

Year	2007	2008	2009	2010	2011	2012	2013
Number of Players	1278	1291	1266	1249	1295	1284	1304

The second part of the question is to find out whether all teams have the same number of players, we still use data from 2000 to 2013. Different from the previous part, we need to group the data by both teamID and yearID using **GROUP BY** operation.

```
SELECT yearID AS year, teamID AS team, COUNT(DISTINCT playerID) AS count
FROM Appearances
WHERE yearID >= '2000'
GROUP BY teamID, yearID
```

We use **ggplot()** to draw the scatter plot, the different color in the plot represents the different year. The plot is shown below:

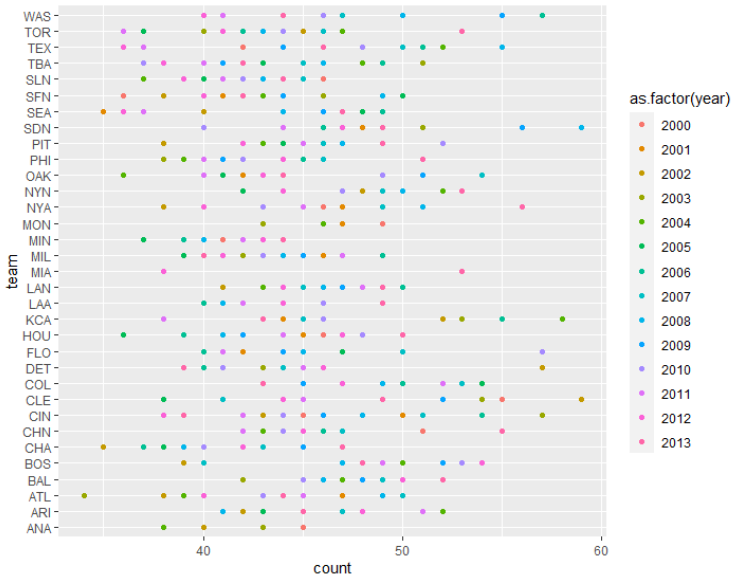

Figure 7: Number of Players in each Team from 2000 to 2013

From the plot we know that for the same team in different year, the number of player is not the same. Also for the different team in the same year, the number of player is still not identical.

To verify both parts in R, we need to first select the yearID greater or equal than 2000 by using **which()** function. Then we will still use functions in the **dplyr** package. This time we will also use the **count()** function to count the number of player.

18 Question 18

Question: ***Do players who hit more home runs receive higher salaries?***
In this question we need to figure out whether players hit more home runs receive higher salaries. Since we cannot directly compare salaries in different year since the value of dollar is different over the years. So we compare the salaries over 3 years. Here we choose 1985 to 1987, 1998 to 2000 and 2011 to 2013 three time periods. Since we want to know the number of home runs, so we need to **LEFT JOIN** the Batting table with Salaries table. For three time periods, the SQL codes are rather similar except the yearID we choose with **WHERE** operation. We also use **ggplot()** to draw scatter plot with smooth line for each time period.

```
SELECT Salaries.yearID, Salaries.salary, Batting.HR
FROM Salaries
LEFT JOIN Batting ON Salaries.yearID = Batting.yearID
                     AND Salaries.playerID = Batting.playerID
                     AND Salaries.teamID = Batting.teamID
WHERE Salaries.yearID >= '1985' AND Salaries.yearID <= '1987'

ggplot(salary_8587_sql, aes(x = HR, y = salary)) +
  geom_point() +
  geom_smooth()

SELECT Salaries.yearID, Salaries.salary, Batting.HR
FROM Salaries
```

```
LEFT JOIN Batting ON Salaries.yearID = Batting.yearID
                     AND Salaries.playerID = Batting.playerID
                     AND Salaries.teamID = Batting.teamID
WHERE Salaries.yearID >= '1998' AND Salaries.yearID <= '2000'

ggplot(salary_9800_sql, aes(x = HR, y = salary)) +
  geom_point() +
  geom_smooth()

SELECT Salaries.yearID, Salaries.salary, Batting.HR
FROM Salaries
LEFT JOIN Batting ON Salaries.yearID = Batting.yearID AND
                     Salaries.playerID = Batting.playerID
                     AND Salaries.teamID = Batting.teamID
WHERE Salaries.yearID >= '2011' AND Salaries.yearID <= '2013'

ggplot(salary_1113_sql, aes(x = HR, y = salary)) +
  geom_point() +
  geom_smooth()
```

The plots for each period are shown below:

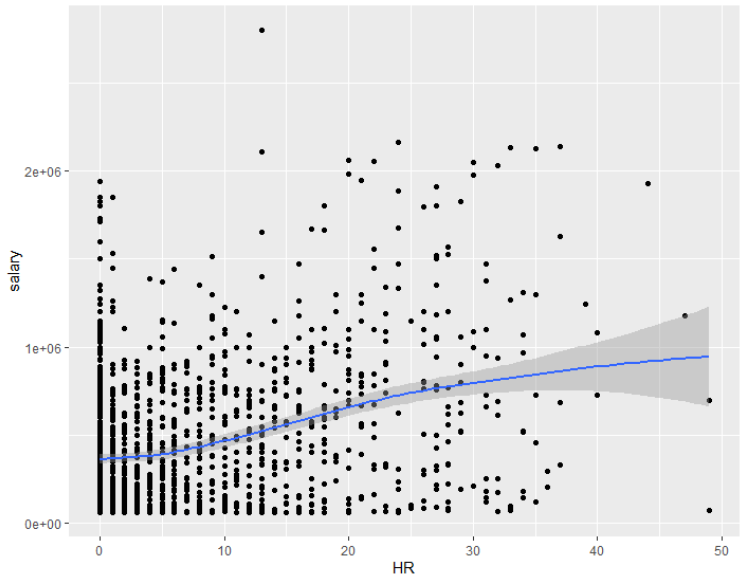

Figure 8: Relationship between Number of Home Runs and Salaries between 1985 and 1987

From the smooth line in the plot we know that during 1985 to 1987, there is a slight increasing trend in salary when the player hit more home runs. However, we cannot say that the more home runs a player hit, the higher salary the player will get, since we cannot observe a clear relationship between two variables.

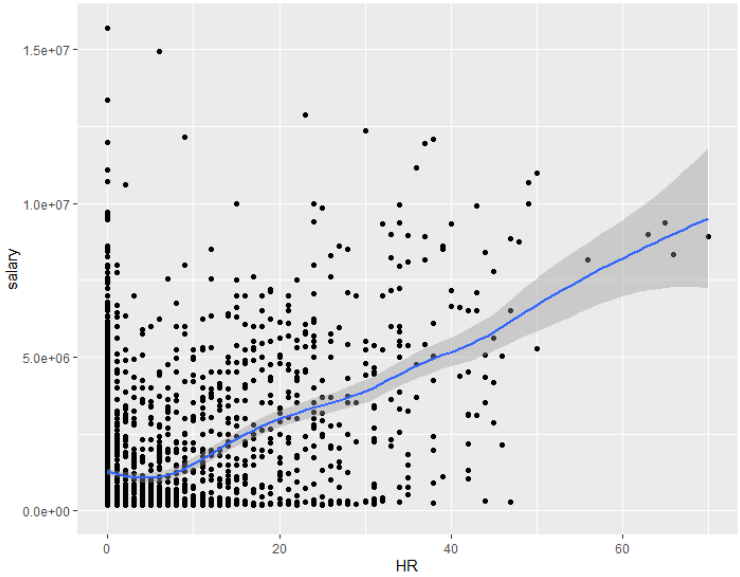

Figure 9: Relationship between Number of Home Runs and Salaries between 1998 and 2000

From the smooth line in the plot we know that during 1998 to 2000, there is an increasing trend in salary when the player hit more home runs. However, we cannot say that the more home runs a player hit, the higher salary the player will get, since we cannot observe a clear relationship between two variables. But if a player hit a large number of hit runs (above 50), his salary will be high.

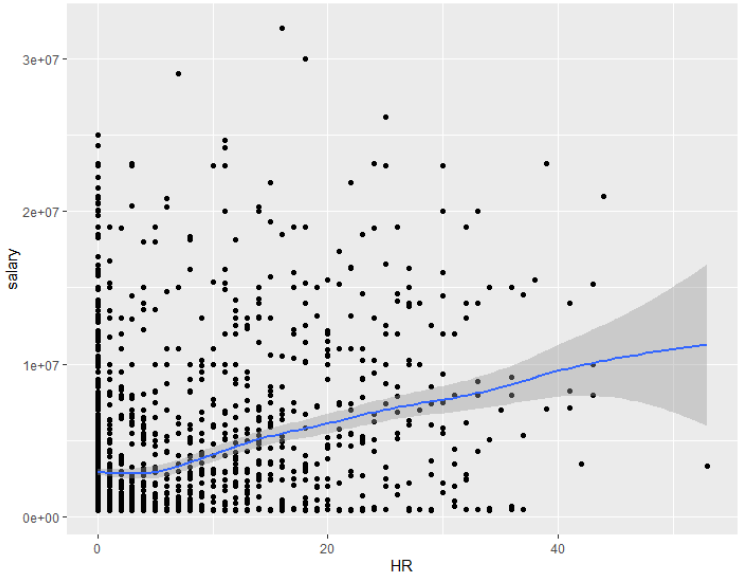

Figure 10: Relationship between Number of Home Runs and Salaries between 2011 and 2013

From the smooth line in the plot we know that during 2011 to 2013, there is a slight increasing trend in salary when the player hit more home runs. However, we still cannot say that the more home runs a player hit, the higher salary the player will get, since we cannot observe a clear relationship between two variables.

Therefore, we cannot say that players will receive higher salaries if they hit more home runs based on the plots above.

To verify in R, we only need to first use **which()** function to select specific time period and use **merge()** function to combine two tables.

```
Batting <- dbGetQuery(baseball, "SELECT * FROM Batting")
Salaries <- dbGetQuery(baseball, "SELECT * FROM Salaries")
Batting_sub1 <- Batting[, c("playerID", "yearID", "teamID", "HR")]
salary_8587_r <- Salaries[which(Salaries$yearID>="1985"&Salaries$yearID<="1987"),]
Merge1 <- merge(salary_8587_r, Batting_sub1,
                by.x = c("playerID", "yearID", "teamID"),
                by.y = c("playerID", "yearID", "teamID"), all.x = TRUE)
ggplot(Merge1, aes(x = HR, y = salary)) +
  geom_point() +
  geom_smooth()

salary_9800_r <- Salaries[which(Salaries$yearID>="1998"&Salaries$yearID<="2000"),]
Merge2 <- merge(salary_9800_r, Batting_sub1,
                by.x = c("playerID", "yearID", "teamID"),
                by.y = c("playerID", "yearID", "teamID"), all.x = TRUE)
ggplot(Merge2, aes(x = HR, y = salary)) +
  geom_point() +
  geom_smooth()

salary_1113_r <- Salaries[which(Salaries$yearID>="2011"&Salaries$yearID<="2013"),]
Merge3 <- merge(salary_1113_r, Batting_sub1,
                by.x = c("playerID", "yearID", "teamID"),
                by.y = c("playerID", "yearID", "teamID"), all.x = TRUE)
ggplot(Merge3, aes(x = HR, y = salary)) +
  geom_point() +
  geom_smooth()
```

19 Question 19

Question: ***Explore the change in salary over time. Use a plot. Identify the teams that won the world series or league on the plot. How does salary relate to winning the league and/or world series.***

In this question we need to plot the change in salary over time, and identify the teams that win the World Series or League Championship. So we first compute the total salary of each team over the years. In this case we need to use the data in Teams table, so we **LEFT JOIN** the Teams table to the Salaries table. Since we need to know the total salaries in each team in each year, we need to group the table by both yearID and teamID using **GROUP BY** operation.

```
SELECT Salaries.yearID, Salaries.teamID, Teams.LgWin, Teams.WSWin,
       SUM(Salaries.salary) AS salary
FROM Salaries
LEFT JOIN Teams ON Salaries.yearID = Teams.yearID
                AND Salaries.teamID = Teams.teamID
                AND Salaries.lgID = Teams.lgID
GROUP BY Salaries.yearID, Salaries.teamID
```

Since the 1994 MLB World Series and the League Championship Series were canceled due to the strike by the MLB Players Association, there was no winner in 1994, and we replace all NAs in 1994 with "N". Then we use **ggplot()** to draw scatter plot, we also use different size and different shape to identify whether a team win the World Series or League Championship.

```
salary_win_sql[is.na(salary_win_sql)] <- "N"
ggplot(salary_win_sql,aes(x=yearID,y=salary,color=teamID,shape=LgWin,size=WSWin)) +
  geom_point() +
  theme(legend.direction = "horizontal", legend.box = "vertical") +
  guides(col = guide_legend(nrow = 12))
```

The plot is shown below:

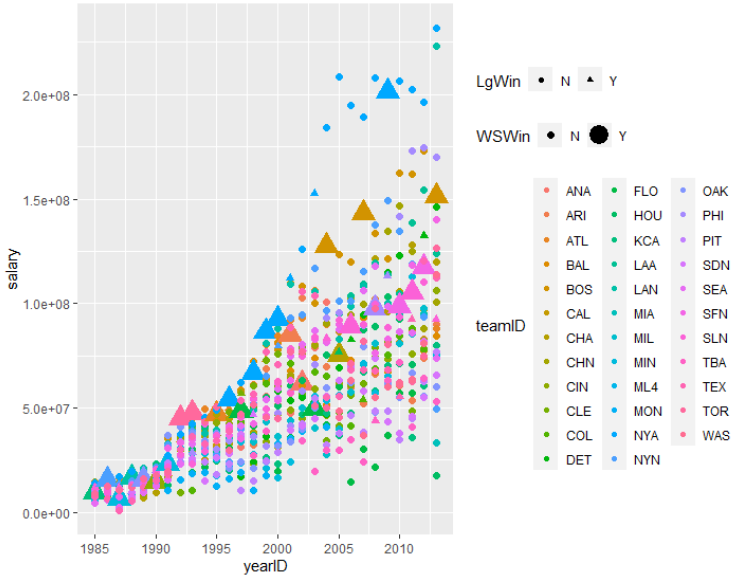

Figure 11: Salaries in Different Team and Year

From the plot we know that in most years, the team won the World Series or/and League Championship always had a higher salary than most of teams. Also we know that as time goes on, the difference of total salary between teams becomes bigger, and the overall salary becomes higher.

To verify it in R, we need to first combine two tables using **merge()** function, then get the total salary in different team and year by using the functions in **dplyr** package. Functions are all discussed in previous questions.

```
Salaries <- dbGetQuery(baseball, "SELECT * FROM Salaries")
Teams <- dbGetQuery(baseball, "SELECT * FROM Teams")
Teams_sub <- Teams[, c("yearID", "teamID", "lgID", "LgWin", "WSWin")]
salary_win_r <- merge(Salaries, Teams_sub,
                      by.x = c("yearID", "teamID", "lgID"),
                      by.y = c("yearID", "teamID", "lgID"), all.x = TRUE)
salary_win_r <- salary_win_r %>%
                  select(yearID, teamID, LgWin, WSWin, salary) %>%
                  group_by(yearID, teamID, LgWin, WSWin) %>%
                  summarise(salary = sum(salary))
salary_win_r[is.na(salary_win_r)] <- "N"
```

```
ggplot(salary_win_r,aes(x=yearID,y=salary,color=teamID,shape=LgWin,size=WSWin)) +
  geom_point() +
  theme(legend.direction = "horizontal", legend.box = "vertical") +
  guides(col = guide_legend(nrow = 12))
```

20 Question 20

Question: ***How many people are there in each categories? For inducted players who were voted by BBWAA (Baseball Writers' Association of America), how many percentage of vote did they receive?***

This question is not listed on the prompt. For this question I want to explore the voting in the Hall of Fame (HoF) since people in the HoF are famous and it is interesting to explore how they come into the HoF. In the first part of the question, I want to know the category of the HoF, and know the number of people in each category. To do this, we need to group the people in the table by category using **GROUP BY** operation, then **COUNT** the number of people in each category.

```
SELECT category, COUNT(category) AS count
FROM HallofFame
GROUP BY category
```

The results are shown in the table below:

Category	Number of People
Manager	74
Pioneer/Executive	39
Player	3931
Umpire	10

From the result we know that most people in the HoF are player. So in the next part of the question, I want to know how much percentage of vote a player receive in order to be inducted in the HoF. Also, we will just explore those players voted by BBWAA (Baseball Writers' Association of America). To do this, We still need to **LEFT JOIN** the Master table to get the first and last name of players, and also make sure we select inducted players voted by BBWAA using **WHERE** operation. To compute the percentage of the vote, we will use the total number of votes received by a player divided by the total ballots cast in that year.

```
SELECT H.playerID AS player, Master.nameFirst AS first_name,
       Master.nameLast AS last_name, H.yearid AS year,
       100.0 * H.votes / H.ballots AS percentage_of_votes_received,
       100.0* H.needed / H.ballots AS percentage_of_votes_needed,
       100.0 * (H.votes-H.needed) / H.ballots AS percentage_exceeded
FROM HallofFame AS H
LEFT JOIN Master ON H.playerID = Master.playerID
WHERE H.votedBy = 'BBWAA' AND H.category = 'Player' AND H.inducted = 'Y'
```

Then I will draw the scatter plot to see the result using **ggplot()**. The red point is the percentage of vote a player received in each year, and the blue point is the percentage of vote a player needed to be inducted in each year.

```
ggplot()+
  geom_point(data = HoF_percentage_sql, aes(x = year,
                                            y = percentage_of_votes_received,
                                            colour = "Received")) +
  geom_point(data = HoF_percentage_sql, aes(x = year,
                                            y = percentage_of_votes_needed,
                                            colour = "Needed")) +
  scale_color_manual(name = "Type of Percentage",
                     values = c(Received = "red", Needed = "blue")) +
  ylab("Percentage of Vote")
```

The plot is shown below:

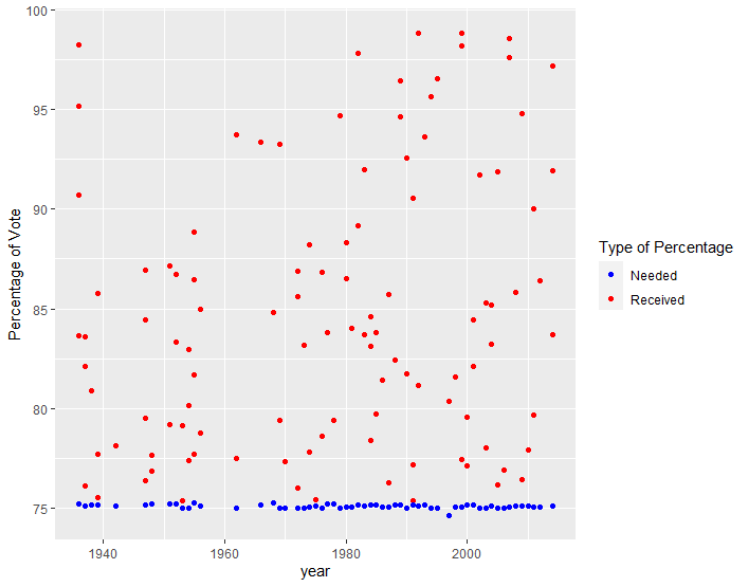

Figure 12: Percentage of Vote in Each Year

From the plot we know that a player need to get at least 75% of the vote to be inducted. Most of the inducted player get 75% to 95% of the vote, and some of the players even get more than 95% of the vote.

To verify the first part in R, we just need to use **table()** function to count the number of people in each category. To verify the second part in R, we need to first **merge()** two tables, and calculate the percentage of vote based on the value in several columns. Then we can use the result to draw scatter plot.

```
HallOfFame <- dbGetQuery(baseball, "SELECT * FROM HallOfFame")
Master <- dbGetQuery(baseball, "SELECT * FROM Master")
table(HallOfFame$category)
sub_master <- Master[, c("playerID", "nameFirst", "nameLast")]
HoF_percentage_r <- merge(HallOfFame, sub_master, by.x = "playerID",
                          by.y = "playerID", all.x = TRUE)
HoF_percentage_r <- HoF_percentage_r[which(HoF_percentage_r$votedBy == "BBWAA"
                                           & HoF_percentage_r$category == "Player"
                                           & HoF_percentage_r$inducted == "Y"), ]
```

```
HoF_percentage_r$percentage_of_votes_received <- 100.0*HoF_percentage_r$votes /
                                                  HoF_percentage_r$ballots
HoF_percentage_r$percentage_of_votes_needed <- 100.0*HoF_percentage_r$needed /
                                                HoF_percentage_r$ballots
HoF_percentage_r$percentage_exceeded <- HoF_percentage_r$percentage_of_votes_received-
                                        HoF_percentage_r$percentage_of_votes_needed
ggplot()+
  geom_point(data = HoF_percentage_r, aes(x = yearid,
                                          y = percentage_of_votes_received,
                                          colour = "Received"))+
  geom_point(data = HoF_percentage_r, aes(x = yearid,
                                          y = percentage_of_votes_needed,
                                          colour = "Needed"))+
  scale_color_manual(name = "Type of Percentage",
                     values = c(Received = "red", Needed = "blue"))+
  xlab("year")+
  ylab("Percentage of Vote")
```