Lab/HW 5: dplyr, SQL

Your lab/homework must be submitted in with two files: (1) R Markdown format file; (2) a pdf or html file. Other formats will not be accepted. Your responses must be supported by both textual explanations and the code you generate to produce your result.

For this homework, we will look trends in baseball team payrolls between the years 1985 and 2010. The data come from the Baseball Databank and is based in part on Lahman's Baseball Database. You will need to download the SQLite database file baseball.db from our course webpage to your computer.

Part I - `dplyr`

The following exercises use the `flights` data set from the `nycflights13` package, you are asked to use `dplyr` functions and verbs. In Both questions you should use the pipe (`%>%`) operator to compute this in one chain.

1. Which destinations are served by at least three airlines? The final answer should include only the destination and the number of carriers. It should be sorted first in a descending order of the number of carriers, and the in ascending order of destination names.

```
library(nycflights13)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
flights <- nycflights13::flights

flights %>%
  group_by(dest) %>%
  summarise(carriers = n_distinct(carrier)) %>%
  filter(carriers >= 6) %>%
  arrange(desc(carriers), dest)
```

```
## # A tibble: 13 x 2
##    dest  carriers
##    <chr>    <int>
##  1 ATL          7
##  2 BOS          7
##  3 CLT          7
##  4 ORD          7
##  5 TPA          7
##  6 AUS          6
##  7 DCA          6
```

```
##  8 DTW          6
##  9 IAD          6
## 10 MSP          6
## 11 MSY          6
## 12 PIT          6
## 13 STL          6
```

2. Which carrier has the highest number of delyaed flights and which the lowest (and what are the corresponding number of delays)? The final output should only contain the airline symbol and number of delays.

```
(has_most_delays <-
   flights %>%
   group_by(carrier) %>%
   filter(dep_delay > 0) %>%
   summarize(n_delay = n()) %>%
   filter(n_delay == max(n_delay) | n_delay == min(n_delay)) %>%
   select(carrier, n_delay))
```

```
## # A tibble: 2 x 2
##   carrier n_delay
##   <chr>     <int>
## 1 OO            9
## 2 UA        27261
```

Part II - SQL

1. Here we will import payroll data from the database.

   a. Using `DBI` and `RSQLite`, setup a connection to the `SQLite` database stored in `baseball.db`. Use `dbListTables()` to list the tables in the database.

```
options(max.print = 60)
library(DBI)
library(RSQLite)
drv = dbDriver("SQLite")
con = dbConnect(drv, dbname = "baseball.db")

dbListTables(con)
```

```
##  [1] "AllstarFull"         "Appearances"         "AwardsManagers"
##  [4] "AwardsPlayers"       "AwardsShareManagers" "AwardsSharePlayers"
##  [7] "Batting"             "BattingPost"         "Fielding"
## [10] "FieldingOF"          "FieldingPost"        "HallOfFame"
## [13] "Managers"            "ManagersHalf"        "Master"
## [16] "Pitching"            "PitchingPost"        "Salaries"
## [19] "Schools"             "SchoolsPlayers"      "SeriesPost"
## [22] "Teams"               "TeamsFranchises"     "TeamsHalf"
## [25] "sqlite_sequence"     "xref_stats"
```

```
dbListFields(con, "Salaries")
```

```
## [1] "yearID"   "teamID"   "lgID"     "playerID" "salary"
```

   b. Use the table that contains salaries and compute the payroll for each team in 2010. Use `dbReadTable()` to grab the entirety of the table, then manipulate using `dplyr` verbs. Which teams had the highest payrolls (that is, sum of all paid salaries)?

```
salaries <- dbReadTable(con, "Salaries")

library(dplyr)
(payroll_summary_dplyr <-
   salaries %>%
   filter(yearID == 2010) %>%
   group_by(teamID) %>%
   summarize(payroll = sum(salary)) %>%
   arrange(desc(payroll)))
```

```
## # A tibble: 30 x 2
##    teamID   payroll
##    <chr>      <dbl>
##  1 NYA    206333389
##  2 BOS    162447333
##  3 CHN    146609000
##  4 PHI    141928379
##  5 NYN    134422942
##  6 DET    122864928
##  7 CHA    105530000
##  8 LAA    104963866
##  9 SFN     98641333
## 10 MIN     97559166
## # ... with 20 more rows
```

c. Repeat the previous step, but now do this using only `dbGetQuery()` and SQL. Are your answers `identical()`? Why or why not? Are their values `all_equal()`?

```
(payroll_summary_SQL <-
    dbGetQuery(con, paste("SELECT teamID, Sum(salary) payroll",
                          "FROM Salaries",
                          "WHERE yearID == 2010",
                          "GROUP BY teamID",
                          "ORDER BY payroll DESC")))
```

```
##    teamID   payroll
## 1     NYA 206333389
## 2     BOS 162447333
## 3     CHN 146609000
## 4     PHI 141928379
## 5     NYN 134422942
## 6     DET 122864928
## 7     CHA 105530000
## 8     LAA 104963866
## 9     SFN  98641333
## 10    MIN  97559166
## 11    LAN  95358016
## 12    SLN  93540751
## 13    HOU  92355500
## 14    SEA  86510000
## 15    ATL  84423666
## 16    COL  84227000
## 17    BAL  81612500
## 18    MIL  81108278
## 19    TBA  71923471
```

```
## 20    CIN  71761542
## 21    KCA  71405210
## 22    TOR  62234000
## 23    WAS  61400000
## 24    CLE  61203966
## 25    ARI  60718166
## 26    FLO  57029719
## 27    OAK  55254900
## 28    TEX  55250544
## 29    SDN  37799300
## 30    PIT  34943000
```

```
identical(payroll_summary_dplyr, payroll_summary_SQL)
```

```
## [1] FALSE
```

```
str(payroll_summary_dplyr)
```

```
## tibble [30 x 2] (S3: tbl_df/tbl/data.frame)
##  $ teamID : chr [1:30] "NYA" "BOS" "CHN" "PHI" ...
##  $ payroll: num [1:30] 2.06e+08 1.62e+08 1.47e+08 1.42e+08 1.34e+08 ...
```

```
str(payroll_summary_SQL)
```

```
## 'data.frame':    30 obs. of  2 variables:
##  $ teamID : chr  "NYA" "BOS" "CHN" "PHI" ...
##  $ payroll: num  2.06e+08 1.62e+08 1.47e+08 1.42e+08 1.34e+08 ...
```

```
all_equal(payroll_summary_dplyr, payroll_summary_SQL)
```

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

d. Repeat again Step b., using this time the `dbplyr` functions. After you show the results, show the SQL query that was used behind the scenes.

```
# Note: you do not need to call `library(dbplyr)`. It is loaded automatically with DBI.
salaries_link <- tbl(con, "salaries")

(payroll_summary_dbplyr <-
   salaries_link %>%
   filter(yearID == 2010) %>%
   group_by(teamID) %>%
   summarize(payroll = sum(salary)) %>%
   arrange(desc(payroll)))
```

```
## Warning: Missing values are always removed in SQL aggregation functions.
## Use `na.rm = TRUE` to silence this warning
## This warning is displayed once every 8 hours.
```

```
## # Source:     SQL [?? x 2]
## # Database:   sqlite 3.39.3 [/home/hhh/classes/UMass/F22_stat535/Homework/hw5/baseball.db]
## # Ordered by: desc(payroll)
##    teamID   payroll
##    <chr>      <dbl>
##  1 NYA    206333389
##  2 BOS    162447333
##  3 CHN    146609000
##  4 PHI    141928379
##  5 NYN    134422942
```

```
##  6 DET    122864928
##  7 CHA    105530000
##  8 LAA    104963866
##  9 SFN     98641333
## 10 MIN     97559166
## # ... with more rows
```

```
payroll_summary_dbplyr %>% show_query()
```

```
## <SQL>
## SELECT `teamID`, SUM(`salary`) AS `payroll`
## FROM `salaries`
## WHERE (`yearID` = 2010.0)
## GROUP BY `teamID`
## ORDER BY `payroll` DESC
```

e. Modify the SQL statement to compute the payroll for each team for each year from 1985 to 2010.

```
dbGetQuery(con, paste("SELECT yearID, teamID, Sum(salary) payroll",
                      "FROM Salaries WHERE yearID >= 1985 AND yearID <= 2010",
                      "GROUP BY teamID, yearID",
                      "ORDER BY teamID, yearID"))
```

```
##    yearID teamID   payroll
## 1    1997    ANA  31135472
## 2    1998    ANA  41281000
## 3    1999    ANA  55388166
## 4    2000    ANA  51464167
## 5    2001    ANA  47535167
## 6    2002    ANA  61721667
## 7    2003    ANA  79031667
## 8    2004    ANA 100534667
## 9    1998    ARI  32347000
## 10   1999    ARI  68703999
## 11   2000    ARI  81027833
## 12   2001    ARI  85082999
## 13   2002    ARI 102819999
## 14   2003    ARI  80657000
## 15   2004    ARI  69780750
## 16   2005    ARI  62329166
## 17   2006    ARI  59684226
## 18   2007    ARI  52067546
## 19   2008    ARI  66202712
## 20   2009    ARI  73516666
##  [ reached 'max' / getOption("max.print") -- omitted 718 rows ]
```

f. Do the same with dbplyr.

```
salaries_link %>%
   group_by(yearID, teamID) %>%
   filter(yearID >= 1985 & yearID <= 2010) %>%
   summarize(payroll = sum(salary)) %>%
   arrange(teamID, yearID)
```

```
## `summarise()` has grouped output by "yearID". You can override using the
## `.groups` argument.
```

```
## # Source:     SQL [?? x 3]
```

```
## # Database:   sqlite 3.39.3 [/home/hhh/classes/UMass/F22_stat535/Homework/hw5/baseball.db]
## # Groups:     yearID
## # Ordered by: teamID, yearID
##    yearID teamID   payroll
##     <int> <chr>      <dbl>
##  1   1997 ANA     31135472
##  2   1998 ANA     41281000
##  3   1999 ANA     55388166
##  4   2000 ANA     51464167
##  5   2001 ANA     47535167
##  6   2002 ANA     61721667
##  7   2003 ANA     79031667
##  8   2004 ANA    100534667
##  9   1998 ARI     32347000
## 10   1999 ARI     68703999
## # ... with more rows
```

2. Write a function that accepts three inputs: minimal total salary, first year and last year. The function returns the number of players whose total salary between the first year and last year (inclusive) exceeded the minimal total salary.

The function should execute a SQL query directly.

The output for (100000000, 1995, 2005) is 15. The output for (200000000, 1995, 2010) is 3.

```
player_minTotSal <- function(minTotSal, minYear, maxYear) {
   query <-
      paste(
         "SELECT COUNT(*) FROM
         (
         SELECT playerID, SUM(salary) AS totalSalary
         FROM Salaries
         WHERE yearID >=", minYear, " AND yearID <= ", maxYear,
         "GROUP BY playerID
         HAVING totalSalary > ", minTotSal, ")"
         )
   res <- dbGetQuery(con, query)
   return(res)
}

player_minTotSal(100000000, 1995, 2005)
```

```
##   COUNT(*)
## 1       15
```

```
player_minTotSal(200000000, 1995, 2010)
```

```
##   COUNT(*)
## 1        3
```

3. Write a function that takes as input a year, a name of a team, and minimal AB value, and returns all playerIDs in the input team that had at least the minimal AB value in the input year.

The function should execute a SQL query directly.

The output for (2010, "CHA", 300) is: konerpa01
kotsama01
vizquom01

pierzaj01
pierrju01
riosal01
quentca01
ramiral03
beckhgo01

```
player_minAB <- function(year, team, minAB) {
   query <-
      paste(
         "SELECT playerID ",
          "FROM BATTING ",
          "WHERE teamID LIKE \"", team , "\" AND yearID = ", year, " AND AB > ", minAB,
         sep = ""
      )
   res <- dbGetQuery(con, query)
   return(res)
}

player_minAB(2010, "CHA", 300)
```

```
##    playerID
## 1 konerpa01
## 2 kotsama01
## 3 vizquom01
## 4 pierzaj01
## 5 pierrju01
## 6  riosal01
## 7 quentca01
## 8 ramiral03
## 9 beckhgo01
```