STA 141B Assignment 3

Suvethika Kandasamy

2024-05-21

These are the tables we are working with for this report.

```
library(RSQLite)
library(DBI)
library(ggplot2)
db <- dbConnect(SQLite(), "stats.stackexchange.db")
dbListTables(db)
```

```
##  [1] "BadgeClassMap"     "Badges"            "CloseReasonMap"
##  [4] "Comments"          "LinkTypeMap"       "PostHistory"
##  [7] "PostHistoryTypeId" "PostLinks"         "PostTypeIdMap"
## [10] "Posts"             "TagPosts"          "Users"
## [13] "VoteTypeMap"       "Votes"
```

Here are some of the fields within these tables: Posts, PostHistory, Comments, Users, Badges

```
dbListFields(db, "Posts")
```

```
##  [1] "Id"                    "PostTypeId"            "AcceptedAnswerId"
##  [4] "CreationDate"          "Score"                 "ViewCount"
##  [7] "Body"                  "OwnerUserId"           "LastActivityDate"
## [10] "Title"                 "Tags"                  "AnswerCount"
## [13] "CommentCount"          "ContentLicense"        "LastEditorDisplayName"
## [16] "LastEditDate"          "LastEditorUserId"      "CommunityOwnedDate"
## [19] "ParentId"              "OwnerDisplayName"      "ClosedDate"
## [22] "FavoriteCount"
```

```
dbListFields(db, "PostHistory")
```

```
##  [1] "Id"                "PostHistoryTypeId" "PostId"
##  [4] "RevisionGUID"      "CreationDate"      "UserId"
##  [7] "Text"              "ContentLicense"    "Comment"
## [10] "UserDisplayName"
```

```
dbListFields(db, "Comments")
```

```
## [1] "Id"              "PostId"          "Score"           "Text"
## [5] "CreationDate"    "UserId"          "ContentLicense"  "UserDisplayName"
```

```
dbListFields(db, "Users")
```

```
##  [1] "Id"             "Reputation"     "CreationDate"   "DisplayName"
##  [5] "LastAccessDate" "WebsiteUrl"     "Location"       "AboutMe"
##  [9] "Views"          "UpVotes"        "DownVotes"      "AccountId"
```

```
dbListFields(db, "Badges")
```

```
## [1] "Id"       "UserId"   "Name"     "Date"     "Class"    "TagBased"
```

1. How many posts are there?

```
qry <- "SELECT count(distinct ID)
        FROM Posts"
dbGetQuery(db, qry)
```

```
##   count(distinct ID)
## 1             405220
```

There are 405220 posts in total. I did this by looking at the unqiue Ids in the posts table. I think I could have done this without the DISTINCT part because it looks like Id is the primary key of that table and each row represents a post.

2. How many posts are there since 2020? (Hint: Convert the CreationDate to a year.)

```
qry <- "SELECT COUNT(DATE(CreationDate))
FROM Posts
WHERE DATE(CreationDate) >= '2020-01-01';
"
dbGetQuery(db, qry)
```

```
##   COUNT(DATE(CreationDate))
## 1                    110949
```

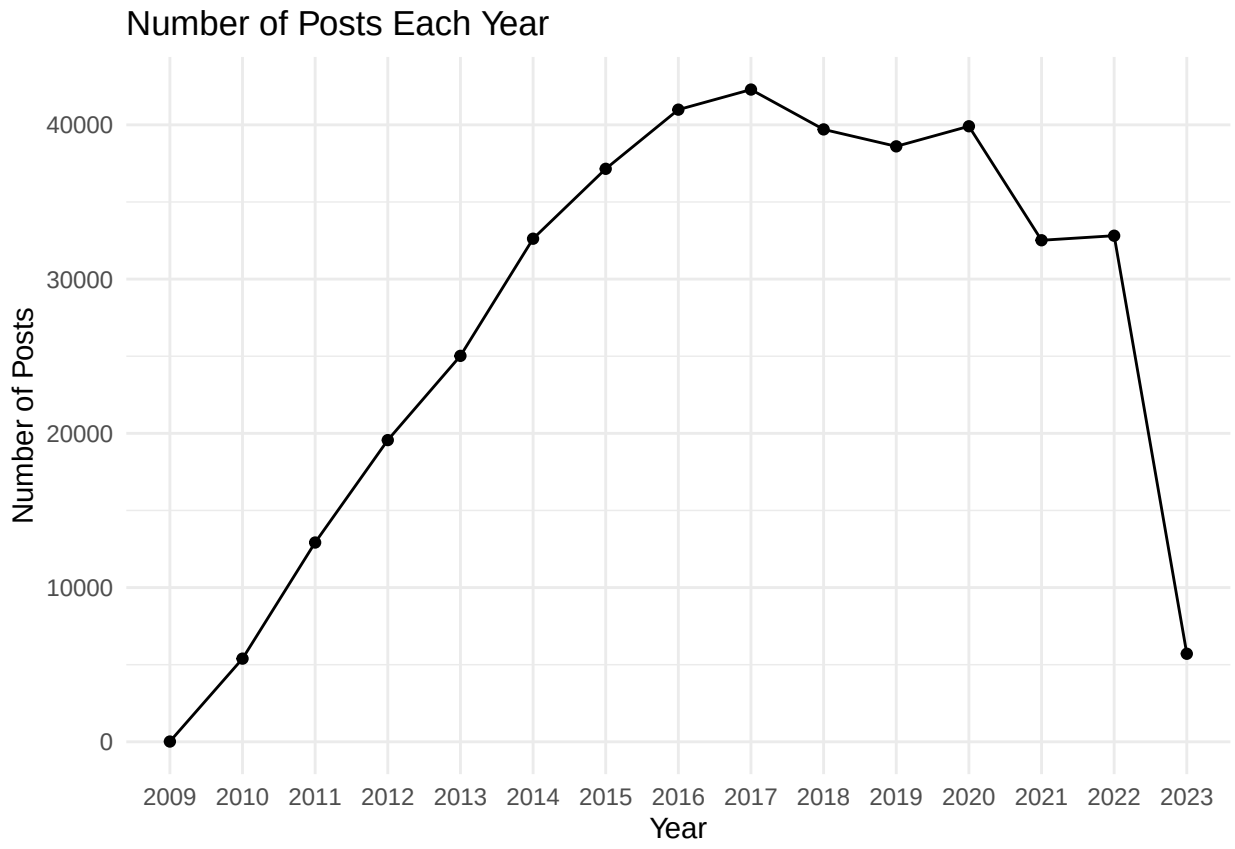
```
 qry <- "SELECT count(CreationDate)
#         FROM Posts;"
#
# dbGetQuery(db, qry)
#
# qry <- "SELECT COUNT(DATE(CreationDate))
# FROM Posts
# WHERE DATE(CreationDate) < '2020-01-01';
# "
# dbGetQuery(db, qry)
```

There are 110949 posts since 2020. I converted the Creation Date and selected the dates that were >= 2020 to get this number. Its adds up to the proper amount (checked how many below 2020 and adss up to the total) and I looked into some the variables inside and they are accurately below 2020 or in 2020 and above.

3. How many posts are there each year? Describe this with a plot, commenting on any anomalies

```
qry <- "SELECT STRFTIME('%Y', CreationDate) AS Year, COUNT(ID) as PostCount
        FROM Posts
        GROUP BY STRFTIME('%Y', CreationDate);"
result <- dbGetQuery(db, qry)

df <- data.frame(result)
ggplot(df, aes(x = Year, y = PostCount)) +
  geom_line(group = 1) +  # Line plot
  geom_point() +          # Points on the line
  labs(title = "Number of Posts Each Year",
       x = "Year",
       y = "Number of Posts") +
  theme_minimal()
```



4. How many tags are in most questions?

```
qry <- "SELECT P.PostTypeId, P.Tags, M.value
        FROM Posts as P
        INNER JOIN PostTypeIdMap as M
        ON P.PostTypeId = M.id
        WHERE M.value = 'Question';"
result <- dbGetQuery(db, qry)

count_tags <- function(tags) {
  if (is.na(tags) || tags == "") {
```

```
    return(0)
  }
  return(length(strsplit(tags, "><")[[1]]))
}

result$TagCount <- sapply(result$Tags, count_tags)

tag_count_table <- table(result$TagCount)
tag_count_table
```

```
##
##     1     2     3     4     5
## 20163 49278 59782 43249 31898
```

```
most_common_tag_count <- as.integer(names(tag_count_table)[which.max(tag_count_table)])
most_common_tag_count
```

```
## [1] 3
```

The most common number of tags for a question is 3.

5. How many posted questions are there?

```
qry <- "SELECT COUNT(P.PostTypeId) as NumofPostedQuestions
        FROM Posts as P
        INNER JOIN PostTypeIdMap as M
        ON P.PostTypeId = M.id
        WHERE M.value = 'Question';"
dbGetQuery(db, qry)
```

```
##   NumofPostedQuestions
## 1               204370
```

There are 204370 posted questions.

6. How many answers are there? (#7)

```
qry <- "SELECT COUNT(P.PostTypeId) as NumofPostedAnswers
        FROM Posts as P
        INNER JOIN PostTypeIdMap as M
        ON P.PostTypeId = M.id
        WHERE M.value = 'Answer';"
dbGetQuery(db, qry)
```

```
##   NumofPostedAnswers
## 1             197928
```

There are 197928 answers.

7. (16.) How many comments are there across all posts? • How many posts have a comment? • What is the distribution of comments per question?

```
qry <- "SELECT COUNT(*) AS numoftotalcomments
        FROM Comments;"
dbGetQuery(db, qry)
```

```
##   numoftotalcomments
## 1             768069
```

```
qry <- "SELECT COUNT(DISTINCT PostId) AS numofpostswithcomments
        FROM Comments;"
dbGetQuery(db, qry)
```

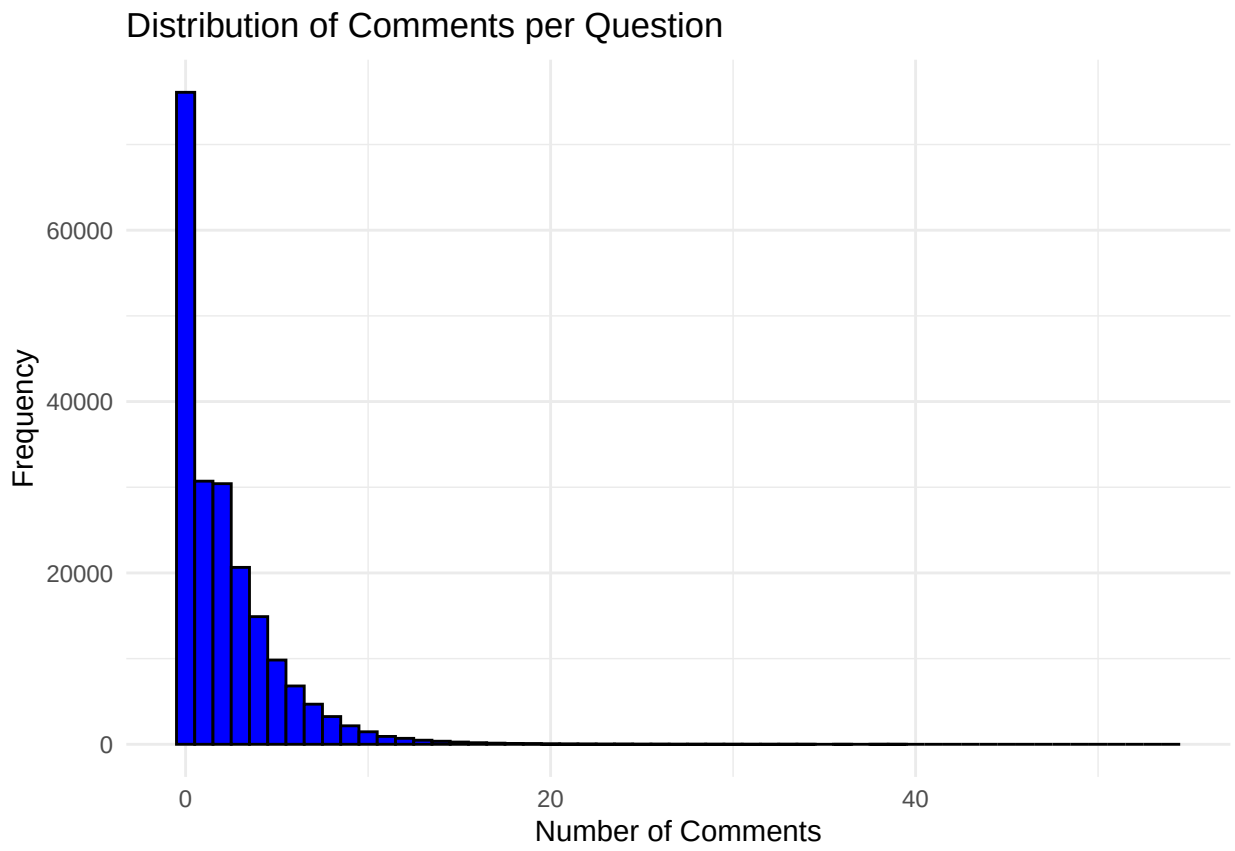
```
##   numofpostswithcomments
## 1                 229859
```

```
qry <- "SELECT CommentCount
        FROM Posts
        WHERE PostTypeId = 1;"
comment_counts <- dbGetQuery(db, qry)

summary(comment_counts$CommentCount)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   1.000   2.171   3.000  54.000
```

```
ggplot(comment_counts, aes(x = CommentCount)) +
  geom_histogram(binwidth = 1, fill = "blue", color = "black") +
  labs(title = "Distribution of Comments per Question",
       x = "Number of Comments",
       y = "Frequency") +
  theme_minimal()
```



8. (19.) How many questions were edited by the original poster? by other users?

```
# qry <- "SELECT COUNT(LastEditorUserId) AS edittedbyoriginalposter
#         FROM Posts
#         WHERE PostTypeId = 1 AND OwnerUserId = LastEditorUserId
#         LIMIT 10;"
# dbGetQuery(db, qry)
#
qry <- "SELECT COUNT(LastEditorUserId) AS edittedbyoriginalposter
        FROM Posts AS P
        INNER JOIN
        PostHistory AS PH
        ON P.Id = PH.PostId
        WHERE P.PostTypeId = 1 AND P.OwnerUserId = LastEditorUserId
        LIMIT 10;"
dbGetQuery(db, qry)
```

```
##   edittedbyoriginalposter
## 1                  302642
```

```
qry <- "SELECT count(LastEditorUserId)
        FROM Posts
        WHERE PostTypeId = 1
        AND LastEditorUserId IS NOT NULL
```

```
        AND LastEditorUserId != ''
        AND LastEditorUserId != OwnerUserId;"
dbGetQuery(db, qry)
```

```
##   count(LastEditorUserId)
## 1                   72582
```

302642 questions were edited by the original poster. 72582 questions were edited by other users. I checked if the numbers add up. Originally checked how many posts had edits in total and then checked if the two numbers added up to the total expected.

9. (20.) How many posts have multiple different people who edit it?

```
qry <- "SELECT COUNT(*)
        FROM (
            SELECT PostId
            FROM PostHistory
            GROUP BY PostId
            HAVING COUNT(DISTINCT UserId) > 1
        ) AS MultiEditorPosts;"
dbGetQuery(db, qry)
```

```
##   COUNT(*)
## 1   143544
```

```
# qry <- "SELECT PostId, COUNT(DISTINCT(UserId))
#         FROM PostHistory
#         GROUP BY PostId
#         HAVING COUNT(DISTINCT(UserId)) > 1;"
# dbGetQuery(db, qry)

# double checking if the code is accurately doing what I want
# qry <- "SELECT UserId
#         FROM PostHistory
#         WHERE PostId = 2;"
# dbGetQuery(db, qry)
#
#
# dbListFields(db, "PostHistory")
```

143544 posts have multiple different people who edit it.

10. (15.) What question has the most comments associated with it? • how many answers are there for this question?

```
qry <- "SELECT C.PostId, Count(C.Id)
        FROM Posts as P
        INNER JOIN Comments as C
        ON P.Id = C.PostId
        WHERE P.PostTypeId = 1
```

```
        HAVING COUNT(C.Id) = MAX(COUNT(C.Id))
        Group by  C.PostId;"

qry <- "SELECT C.PostId, COUNT(C.Id) AS CommentCount, P.AnswerCount
        FROM Posts AS P
        INNER JOIN Comments AS C ON P.Id = C.PostId
        WHERE P.PostTypeId = 1
        GROUP BY C.PostId
        HAVING COUNT(C.Id) = (
            SELECT MAX(CommentCount)
            FROM (
                SELECT COUNT(C.Id) AS CommentCount
                FROM Posts AS P
                INNER JOIN Comments AS C ON P.Id = C.PostId
                WHERE P.PostTypeId = 1
                GROUP BY C.PostId
            ) AS CommentCounts
        );"
dbGetQuery(db, qry)
```

```
##   PostId CommentCount AnswerCount
## 1 328630           54           6
```

Required Questions

21. Compute the table that contains • the question, • the name of the user who posted it, • when that user joined, • their location • the date the question was first posted, • the accepted answer, • when the accepted answer was posted • the name of the user who provided the accepted answer.

```
qry <- "
  SELECT
    Q.Body AS Question,
    U.DisplayName AS UserName,
    U.CreationDate AS UserJoinDate,
    U.Location AS UserLocation,
    Q.CreationDate AS PostCreationDate,
    Q.AcceptedAnswerId,
    A.Body AS AcceptedAnswer,
    A.CreationDate AS AcceptedAnswerDate,
    AU.DisplayName AS AcceptedAnswerUserName
  FROM
    Posts AS Q
  INNER JOIN
    Users AS U ON Q.OwnerUserId = U.Id
  LEFT JOIN
    Posts AS A ON Q.AcceptedAnswerId = A.Id
  LEFT JOIN
    Users AS AU ON A.OwnerUserId = AU.Id
  WHERE
    Q.PostTypeId = 1;"

result <- dbGetQuery(db, qry)
```

```
## Warning: Column `AcceptedAnswerId`: mixed type, first seen values of type
## integer, coercing other values of type string
```

```
qry <- "
  SELECT
    Q.Body AS Question,
    U.DisplayName AS UserName,
    U.CreationDate AS UserJoinDate,
    U.Location AS UserLocation,
    Q.CreationDate AS PostCreationDate,
    Q.AcceptedAnswerId,
    A.Body AS AcceptedAnswer,
    A.CreationDate AS AcceptedAnswerDate,
    AU.DisplayName AS AcceptedAnswerUserName
  FROM
    Posts AS Q
  INNER JOIN
    Users AS U ON Q.OwnerUserId = U.Id
  LEFT JOIN
    Posts AS A ON Q.AcceptedAnswerId = A.Id
  LEFT JOIN
    Users AS AU ON A.OwnerUserId = AU.Id
  WHERE
    Q.PostTypeId = 1
  LIMIT 10;"

dbGetQuery(db, qry)
```

```
## Warning: Column `AcceptedAnswerId`: mixed type, first seen values of type
## integer, coercing other values of type string
```

```
##
## 1
## 2
## 3
## 4
## 5  <p>Last year, I read a blog post from <a href="http://anyall.org/">Brendan O'Connor</a> entitled
## 6
## 7
## 8
## 9
## 10
##               UserName            UserJoinDate                 UserLocation
## 1          csgillespie 2010-07-19T19:04:52.280   Newcastle, United Kingdom
## 2              A Lion 2010-07-19T19:09:32.157
## 3              grokus 2010-07-19T19:08:29.070               United States
## 4          Jay Stevens 2010-07-19T19:09:16.917       Jacksonville, FL, USA
## 5               Shane 2010-07-19T19:03:57.227                New York, NY
## 6              EAMann 2010-07-19T19:11:57.393 Tualatin, OR, United States
## 7              A Lion 2010-07-19T19:09:32.157
## 8  Christopher D. Long 2010-07-19T19:11:11.093              Versailles, KY
## 9            dassouki 2010-07-19T19:19:26.317         Fredericton, Canada
## 10     Daniel Vassallo 2010-07-19T19:21:06.623                 Seattle, WA
```

```
##           PostCreationDate AcceptedAnswerId
## 1  2010-07-19T19:12:12.510               15
## 2  2010-07-19T19:12:57.157               59
## 3  2010-07-19T19:13:28.577                5
## 4  2010-07-19T19:13:31.617              135
## 5  2010-07-19T19:14:44.080                0
## 6  2010-07-19T19:15:59.303               18
## 7  2010-07-19T19:17:47.537             1887
## 8  2010-07-19T19:18:30.810             1201
## 9  2010-07-19T19:24:36.303                0
## 10 2010-07-19T19:25:39.467                0
##
## 1
## 2
## 3
## 4
## 5
## 6
## 7  <p>Maybe too late but I add my answer anyway...</p>\\\\n\\\\n<p>It depends on what you intend to
## 8
## 9
## 10
##         AcceptedAnswerDate AcceptedAnswerUserName
## 1  2010-07-19T19:19:46.160                 Harlan
## 2  2010-07-19T19:43:20.423         John L. Taylor
## 3  2010-07-19T19:14:43.050            Jay Stevens
## 4  2010-07-19T21:36:12.850         John L. Taylor
## 5                      <NA>                   <NA>
## 6  2010-07-19T19:24:18.580         Stephen Turner
## 7  2010-08-19T10:00:00.370                    chl
## 8  2010-08-03T21:50:09.007         Carlos Accioly
## 9                      <NA>                   <NA>
## 10                     <NA>                   <NA>
```

22. Determine the users that have only posted questions and never answered a question? (Compute the table containing the number of questions, number of answers and the user's login name for this group.) How many are there?

```
qry <- "
  SELECT
    U.DisplayName AS UserName,
    Q.NumQuestions,
    COALESCE(A.NumAnswers, 0) AS NumAnswers
  FROM
    Users AS U
  LEFT JOIN (
    SELECT
      OwnerUserId,
      COUNT(*) AS NumQuestions
    FROM
      Posts
    WHERE
      PostTypeId = 1
    GROUP BY
```

```
      OwnerUserId
  ) AS Q ON U.Id = Q.OwnerUserId
  LEFT JOIN (
    SELECT
      OwnerUserId,
      COUNT(*) AS NumAnswers
    FROM
      Posts
    WHERE
      PostTypeId = 2
    GROUP BY
      OwnerUserId
  ) AS A ON U.Id = A.OwnerUserId
  WHERE
    Q.NumQuestions IS NOT NULL
    AND COALESCE(A.NumAnswers, 0) = 0;
"
result <- dbGetQuery(db, qry)

qry <- "
  SELECT
    U.DisplayName AS UserName,
    Q.NumQuestions,
    COALESCE(A.NumAnswers, 0) AS NumAnswers
  FROM
    Users AS U
  LEFT JOIN (
    SELECT
      OwnerUserId,
      COUNT(*) AS NumQuestions
    FROM
      Posts
    WHERE
      PostTypeId = 1
    GROUP BY
      OwnerUserId
  ) AS Q ON U.Id = Q.OwnerUserId
  LEFT JOIN (
    SELECT
      OwnerUserId,
      COUNT(*) AS NumAnswers
    FROM
      Posts
    WHERE
      PostTypeId = 2
    GROUP BY
      OwnerUserId
  ) AS A ON U.Id = A.OwnerUserId
  WHERE
    Q.NumQuestions IS NOT NULL
    AND COALESCE(A.NumAnswers, 0) = 0
  LIMIT 10;
"
```

```
dbGetQuery(db, qry)
```

```
##           UserName NumQuestions NumAnswers
## 1           grokus            2          0
## 2           A Lion            2          0
## 3           EAMann            1          0
## 4          Alan H.           13          0
## 5             kyle            2          0
## 6           Preets            1          0
## 7           Martin            2          0
## 8  Daniel Vassallo            1          0
## 9     Oren Hizkiya            3          0
## 10           bshor            1          0
```

There are 76077+333 users in total that asked questions but never answered any.

23. Compute the table with information for the 75 users with the most accepted answers. This table should include • the user's display name, • *creation date,* • location, • *the number of badges they have won,* – the names of the badges (as a single string) • *the dates of the earliest and most recent accepted answer (as two fields)* – the (unique) tags for all the questions for which they had the accepted answer (as a single string)

I did 26 on a seperate file because it is using up too much memory for the R markdown to knit.

```
# qry <- "SELECT OwnerUserId, COUNT(*) AS numofanswers
# FROM Posts
# WHERE Id in (SELECT AcceptedAnswerId
# FROM Posts
# WHERE PostTypeId = 1 AND AcceptedAnswerId != '')
# GROUP By OwnerUserId
# ORDER BY COUNT(*) DESC
# LIMIT 75;
# "
#
# qry <- "SELECT Id, AcceptedAnswerId, Tags
# FROM Posts
# WHERE PostTypeId = 1 AND AcceptedAnswerId != '';
# "
#
# qry <- "SELECT OwnerUserId, MIN(CreationDate) AS EarliestAcceptedAnsDate, MAX(CreationDate) AS Recent
# FROM Posts
# WHERE Id in (SELECT AcceptedAnswerId
# FROM Posts
# WHERE PostTypeId = 1 AND AcceptedAnswerId != '')
# GROUP By OwnerUserId;"


# qry <- " SELECT U.Id, B.UserId, U.DisplayName, U.CreationDate AS UserCreationDate, U.Location, COUNT(
# FROM Users AS U
# INNER JOIN Badges AS B
# ON U.Id = B.UserId
# LEFT JOIN (SELECT OwnerUserId, MIN(CreationDate) AS EarliestAcceptedAnsDate, MAX(CreationDate) AS Rec
```

```
# FROM Posts
# WHERE Id in (SELECT AcceptedAnswerId
# FROM Posts
# WHERE PostTypeId = 1 AND AcceptedAnswerId != '')
# GROUP By OwnerUserId) AS A ON U.Id = A.OwnerUserId
# INNER JOIN (
#     SELECT p1.OwnerUserId, GROUP_CONCAT(p2.Tags) AS Tags
#     FROM Posts AS p1
#     INNER JOIN Posts AS p2 ON p1.ParentId = p2.Id
#     WHERE p1.Id IN (SELECT AcceptedAnswerId FROM Posts WHERE PostTypeId = 1 AND AcceptedAnswerId != '
#     GROUP BY p1.OwnerUserId
# ) AS R ON U.Id = R.OwnerUserId
# WHERE U.Id in
# (SELECT OwnerUserId
# FROM Posts
# WHERE Id in (SELECT AcceptedAnswerId
# FROM Posts
# WHERE PostTypeId = 1 AND AcceptedAnswerId != '')
# GROUP By OwnerUserId
# ORDER BY COUNT(*) DESC
# LIMIT 75)
# GROUP BY B.UserId;
# "
# result <- dbGetQuery(db, qry)
#
# clean_tags <- function(tags_string) {
#   tags <- unlist(strsplit(tags_string, ","))
#   unique_tags <- unique(tags)
#   cleaned_tags <- gsub("><", ",", unique_tags)
#   cleaned_tags <- gsub("[<>]", "", cleaned_tags)
#   return(paste(cleaned_tags, collapse = ","))
# }
#
# unique_tags <- function(tags_string) {
#   tags_list <- unlist(strsplit(tags_string, ","))
#   unique_tags <- unique(tags_list)
#   unique_tags_string <- paste(unique_tags, collapse = ", ")
#   return(unique_tags_string)
# }
#
# for (i in 1:nrow(result)) {
#   result[i, "Tags"] <- unique_tags(clean_tags(result[i, "Tags"]))
# }
#
# head(result, 10)
```

24. How many questions received no answers (accepted or unaccepted)? How many questions had no accepted answer?

```
qry_no_answers <- "
SELECT COUNT(*) AS NoAnswers
FROM Posts
WHERE PostTypeId = 1
```

```
AND AnswerCount = 0 OR AcceptedAnswerId = '';
"
dbGetQuery(db, qry_no_answers)
```

```
##   NoAnswers
## 1    337215
```

```
qry_no_accepted_answers <- "
SELECT COUNT(*) AS NoAcceptedAnswers
FROM Posts
WHERE PostTypeId = 1
AND AcceptedAnswerId = '';
"
dbGetQuery(db, qry_no_accepted_answers)
```

```
##   NoAcceptedAnswers
## 1            136365
```

25. What is the distribution of answers per posted question?

```
qry_answers_distribution <- "
SELECT AnswerCount, COUNT(*) AS QuestionCount
FROM Posts
WHERE PostTypeId = 1
GROUP BY AnswerCount
ORDER BY AnswerCount;
"
result_answers_distribution<-dbGetQuery(db, qry_answers_distribution)
result_answers_distribution
```

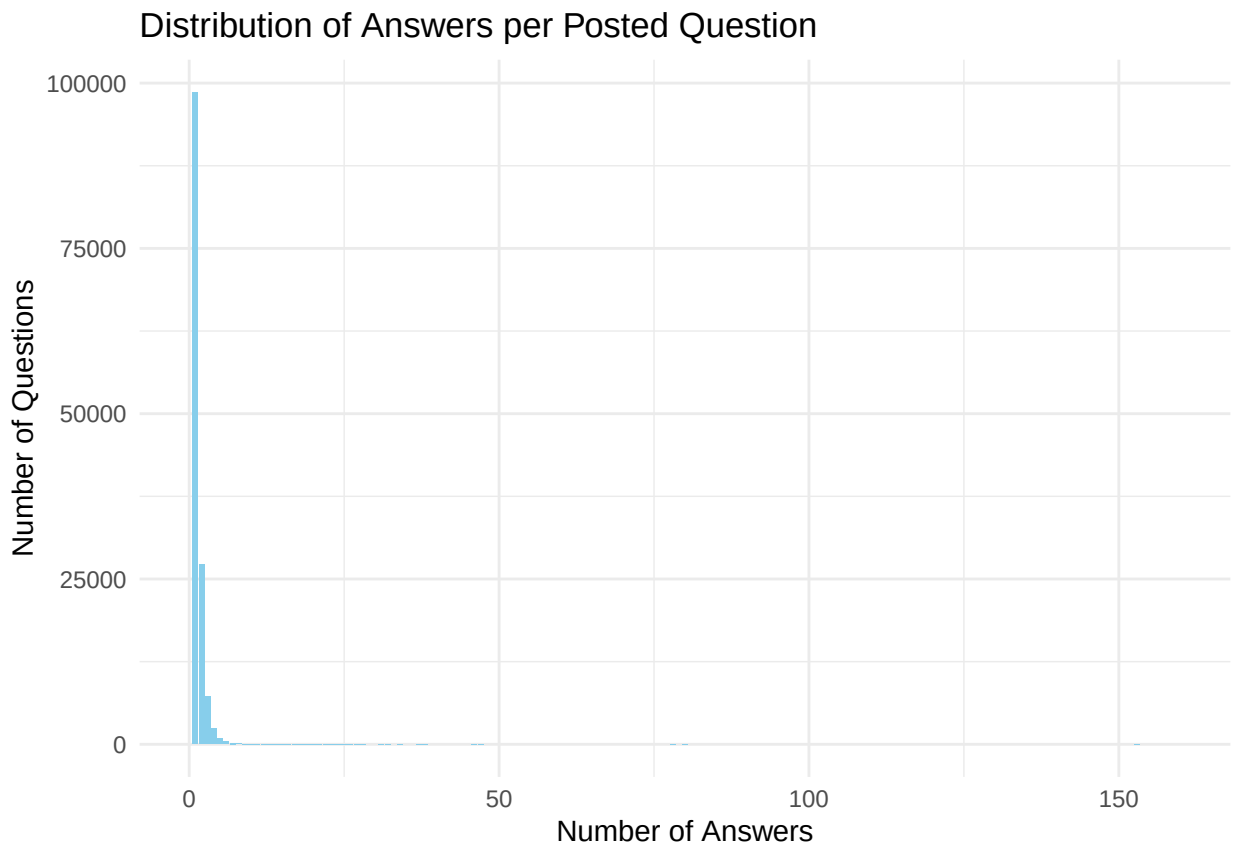
```
##    AnswerCount QuestionCount
## 1            0         66970
## 2            1         98602
## 3            2         27191
## 4            3          7246
## 5            4          2408
## 6            5           905
## 7            6           401
## 8            7           210
## 9            8           136
## 10           9            82
## 11          10            62
## 12          11            35
## 13          12            25
## 14          13            19
## 15          14            14
## 16          15            14
## 17          16             8
## 18          17             5
## 19          18             4
## 20          19             5
```

```
## 21          20             1
## 22          21             5
## 23          22             2
## 24          23             1
## 25          24             1
## 26          25             2
## 27          26             2
## 28          27             1
## 29          28             2
## 30          31             1
## 31          32             2
## 32          34             1
## 33          37             1
## 34          38             1
## 35          46             1
## 36          47             1
## 37          78             1
## 38          80             1
## 39         153             1
```

```r
df <- as.data.frame(result_answers_distribution)

# Plotting the distribution
ggplot(df, aes(x = AnswerCount, y = QuestionCount)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(
    title = "Distribution of Answers per Posted Question",
    x = "Number of Answers",
    y = "Number of Questions"
  ) +
  theme_minimal()+
  xlim(0, 160)
```

```
## Warning: Removed 1 rows containing missing values (`geom_bar()`).
```



26. What is the length of time for a question to receive an answer? to obtaining an accepted answer?

```
qry = "SELECT q.Id AS QId, q.CreationDate AS QDate, a.Id AS AId, MIN(a.CreationDate) AS FirstADate,   ju
                        FROM Posts q
                        JOIN Posts a on QId = a.ParentId AND a.PostTypeId = 2
                        WHERE q.PostTypeId = 1
                        GROUP BY QId, QDate;"
time_data <- dbGetQuery(db, qry)
time_data <- time_data[time_data$ADays >= 0,]
head(time_data, 10)
```

```
##    QId                   QDate  AId              FirstADate        ADays
## 1    1 2010-07-19T19:12:12.510   15 2010-07-19T19:19:46.160 5.250579e-03
## 2    2 2010-07-19T19:12:57.157   20 2010-07-19T19:24:35.803 8.086181e-03
## 3    3 2010-07-19T19:13:28.577    5 2010-07-19T19:14:43.050 8.619558e-04
## 4    4 2010-07-19T19:13:31.617  133 2010-07-19T21:31:53.813 9.609023e-02
## 5    6 2010-07-19T19:14:44.080   13 2010-07-19T19:18:56.800 2.925000e-03
## 6    7 2010-07-19T19:15:59.303   12 2010-07-19T19:18:41.370 1.875775e-03
## 7   10 2010-07-19T19:17:47.537  153 2010-07-19T22:39:27.230 1.400427e-01
## 8   11 2010-07-19T19:18:30.810 1201 2010-08-03T21:50:09.007 1.510530e+01
## 9   17 2010-07-19T19:24:12.187   29 2010-07-19T19:28:15.640 2.817743e-03
## 10  21 2010-07-19T19:24:36.303 2195 2010-08-29T20:01:34.300 4.102567e+01
```

```
mean(time_data$ADays)
```

```
## [1] 54.45071
```

```
median(time_data$ADays)
```

```
## [1] 0.1295946
```

There were some negative values in the answer day column which would mean there are some answers posted before the question? This must be some sort of DB error. After filter those out and then calculating the averages in R: 54.45071 is the avg.

```
qry <- "SELECT
        q.Id AS QId,
        q.CreationDate AS QDate,
        MIN(a.CreationDate) AS FirstADate,
        julianday(MIN(a.CreationDate)) - julianday(q.CreationDate) AS ADays
    FROM
        Posts q
        JOIN Posts a ON q.AcceptedAnswerId = a.Id
    WHERE
        q.PostTypeId = 1
    GROUP BY
        QId, QDate;
"
time_data2 <- dbGetQuery(db, qry)
time_data2 <- time_data2[time_data2$ADays >= 0,]
mean(time_data2$ADays)
```

```
## [1] 25.30695
```

```
median(time_data2$ADays)
```

```
## [1] 0.1410858
```