Stack Exchange DB

Robert Baranic

```
library(RSQLite)
```

```
list.files()
```

```
## [1] "StackExchange.pdf"      "StackExchangeDB.pdf"    "StackExchangeDB.Rmd"
## [4] "stats.stackexchange.db"
```

```
con = dbConnect(SQLite(), "stats.stackexchange.db")
```

```
sql = function(query) dbGetQuery(con, query)
```

```
dbListTables(con)
```

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

1. How many users are there? To do this, we will select the users table and sum the rows since the ID is unique to each user.

```
q = "SELECT COUNT(*)
FROM Users"
sql(q)
```

```
##   COUNT(*)
## 1   321677
```

2. How many users joined since 2020? (Hint: Convert the CreationDate to a year.) To do this, we will convert the CreationDate column to dates and subset the data to >2020. The dates look like this: 2010-07-19T06:55:26.860, so we only need to look at the first 4 characters.

```
q = "SELECT CAST(substr(CreationDate, 1, 4) as decimal) as x
FROM Users
WHERE x >= 2020"
date.df = sql(q)

table(date.df$x)
```

```
##
##  2020  2021  2022  2023
## 34617 32765 28801  4613
```

```r
nrow(date.df)
```

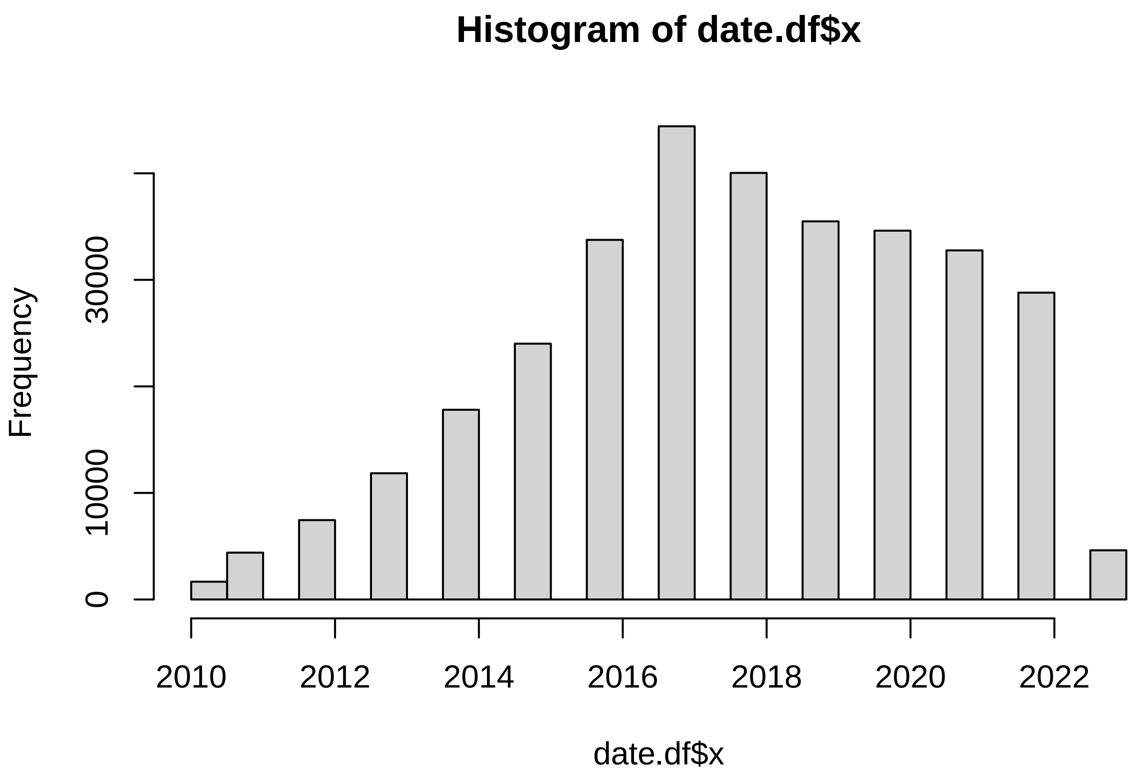
```
## [1] 100796
```

3. How many users joined each year? Describe this with a plot, commenting on any anomalies. We will do this in the same way as before, except we will not subset the data, then we will graph the results. Creation date peaked in 2017 and has been in decline since then. 2023 has very small number of creation date, likely due to it being the current year.

```r
q = "SELECT CAST(substr(CreationDate, 1, 4) as decimal) as x
FROM Users"
date.df = sql(q)

table(date.df$x)
```

```
##
##  2010  2011  2012  2013  2014  2015  2016  2017  2018  2019  2020  2021  2022
##  1668  4396  7450 11846 17809 24012 33753 44416 40040 35491 34617 32765 28801
##  2023
##  4613
```

```r
hist(date.df$x)
```



4. How many different types of posts are there in the Posts table? Get the description of the types from the PostTypeIdMap table. In other words, create a table with the description of each post type and the number of posts of that type, and arrange it from most to least occurrences. Here we will be doing an inner join on the Posts and PostTypeIdMap, joining on their key. Then, we group by PostTypeID and COUNT(*) so we get a sum of the number of those posts. The only other column in PostTypeIdMap is value, so we will join that to the table which contains the description of the post.

```
q = "SELECT PostTypeId, COUNT(*)
FROM Posts
GROUP BY PostTypeId;"

q = "SELECT PostTypeId, Count(*), value
FROM Posts as x
INNER JOIN PostTypeIdMap as y
ON x.PostTypeId = y.id
GROUP BY PostTypeId
ORDER BY COUNT(*) DESC"
sql(q)
```

```
##   PostTypeId Count(*)
## 1          1   204370
## 2          2   197928
## 3          5     1444
## 4          4     1444
## 5          6       23
## 6          3        6
## 7          7        5
##                                                       value
## 1                                                  Question
## 2                                                    Answer
## 3                                                  Tag wiki
## 4                                          Tag wiki excerpt
## 5                                      Moderator nomination
## 6                                          Orphaned tag wiki
## 7 Wiki placeholder (seems to only be the election description)
```

5. How many posted questions are there? From the previous question, we can see that the PostTypeId for "question" is 1, so we can simply subset Posts by this.

```
q = "SELECT COUNT(*)
FROM Posts
WHERE PostTypeId = 1"
sql(q)
```

```
##   COUNT(*)
## 1   204370
```

6. What are the top 50 most common tags on questions? For each of the top 50 tags on questions, how many questions are there for each tag. This will be a similar format to #4. We will group Posts by Tags, COUNT(*) and subset where PostId is 1, so it is a question. Then we will order by desc, limit 50, so we get the top 50.

```
q = "SELECT Tags, COUNT(*)
FROM Posts
WHERE PostTypeId = 1
GROUP BY Tags
ORDER BY COUNT(*) DESC
LIMIT 50"
sql(q)
```

```
##                                               Tags COUNT(*)
## 1                                      <regression>     1019
## 2                                     <probability>      951
## 3                              <hypothesis-testing>      600
## 4                                <machine-learning>      435
## 5                                               <r>      407
## 6                                     <time-series>      398
## 7                                     <correlation>      364
## 8                                   <r><regression>      363
## 9                                 <neural-networks>      362
## 10              <machine-learning><neural-networks>      324
## 11                            <confidence-interval>      290
## 12                        <probability><self-study>      284
## 13                             <data-visualization>      273
## 14                          <regression><logistic>      272
## 15                                     <clustering>      265
## 16                                            <pca>      250
## 17                        <mathematical-statistics>      241
## 18                                  <distributions>      234
## 19                       <statistical-significance>      233
## 20                                       <bayesian>      231
## 21                     <probability><distributions>      227
## 22                                          <anova>      205
## 23                            <normal-distribution>      195
## 24           <probability><conditional-probability>      193
## 25                               <cross-validation>      193
## 26                            <multiple-regression>      191
## 27                <regression><multiple-regression>      181
## 28                                       <sampling>      176
## 29                       <time-series><forecasting>      172
## 30 <hypothesis-testing><statistical-significance>      171
## 31                         <reinforcement-learning>      169
## 32                                 <r><time-series>      166
## 33                      <r><mixed-model><lme4-nlme>      161
## 34                                         <t-test>      160
## 35                                    <mixed-model>      156
## 36                                       <survival>      152
## 37               <machine-learning><classification>      151
## 38                            <clustering><k-means>      141
## 39                                       <logistic>      139
## 40                                 <classification>      139
## 41                             <time-series><arima>      138
## 42                                     <self-study>      138
## 43                                  <meta-analysis>      131
## 44                               <chi-squared-test>      128
## 45                      <regression><correlation>      125
## 46                                          <svm>      123
## 47                           <standard-deviation>      123
## 48                               <markov-process>      123
## 49                                     <r><anova>      122
## 50          <machine-learning><cross-validation>      120
```

8. How many answers are there? Like #5, we will use the same form, but use the answer index of 2 instead of 1.

```
q = "SELECT COUNT(*)
FROM Posts
WHERE PostTypeId = 2"
sql(q)
```

```
##   COUNT(*)
## 1   197928
```

9. What's the most recent question (by date-time) in the Posts table? • Find it on the stats.exchange.com Web site and provide the URL. • How would we map a question in the Posts table to the corresponding SO URL? I understood this as the most recent post in terms of creation date, so we will use this column to determine this. We will convert the characters to a date using datetime() and order by DESC to get the most recent post. Then, we will also include the PostId and the title so we can locate the post to get the URL.

```
q = "SELECT Id, Title, datetime(CreationDate) as Date
FROM Posts
WHERE PostTypeId = 1
ORDER BY Date DESC"

post.df = sql(q)

head(post.df)
```

```
##       Id
## 1 608405
## 2 608403
## 3 608401
## 4 608400
## 5 608398
## 6 608397
##
## 1                                       Are there any papers or methods that combine mcmc and variati
## 2           An example of MA($\\infty$) process with the property of long-term dependence and strictl
## 3                                           What sensitivity analysis technique is suitable for nomin
## 4                                             How sample size affect the chance to reject null hypoth
## 5                                  Is longitudinal data a necessity of mediation paths in structural equa
## 6 Should you impute missing event and time-to-event variables for survival data that has missing and
##                  Date
## 1 2023-03-05 05:10:18
## 2 2023-03-05 04:33:18
## 3 2023-03-05 03:21:49
## 4 2023-03-05 02:51:55
## 5 2023-03-05 01:37:48
## 6 2023-03-05 01:23:04
```

The most recent post was not able to be found, but the second post has the URL as https://stats.stackexchange.com/questions/608403/an-example-of-ma-infty-process-with-the-property-of-long-term-dependence-and

We can see a clear pattern as https://stats.stackexchange.com/questions/%22Id%22/%22title-sepaarated-by-dashes" but the title in the URL cuts off at some point. Based on nchar, the title is 76 characters with the dashes, which could be the limit. We could map a question from the table to the URL by having https://stats.stackexchange.com/questions/ followed by the Id/ and the title with " " replaced by "-" limited to 76 characters, that will get the URL.

```
nchar("an-example-of-ma-infty-process-with-the-property-of-long-term-dependence-and")
```

```
## [1] 76
```

10. For the 10 users who posted the most questions • How many questions did they post? • What are the users' names? • When did they join SO? • What is their Reputation? • What country do they have in their profile? To do this, we will have to join the Posts table with the Users table. We will have to count the number of questions posted by each user and then sort it by DESC. We will join them by matching the Users.Id to Posts.OwnerUserId.

```
q = "SELECT OwnerUserId, DisplayName, u.CreationDate, u.Reputation, u.Location,
 ↪  COUNT(DISTINCT p.Id) as Questions
FROM Posts as p, Users as u
WHERE p.OwnerUserId = u.Id
AND PostTypeId = 1
GROUP BY OwnerUserId
ORDER BY Questions DESC
LIMIT 10"

sql(q)
```

```
##    OwnerUserId                  DisplayName            CreationDate Reputation
## 1        77179                   stats_noob 2015-05-14T21:12:31.790          1
## 2         1005                          Tim 2010-08-19T15:31:09.537      18497
## 3         9162 user1205901 -          2012-02-13T02:09:08.377      11859
## 4        53690                Richard Hardy 2014-08-08T10:57:13.613      60742
## 5       108150                   user321627 2016-03-10T14:45:28.010       2478
## 6       113777                    Haitao Du 2016-04-27T20:51:38.203      34665
## 7        28986               Charlie Parker 2013-08-09T19:20:37.540       6286
## 8        40252         An old man in the sea. 2014-02-14T13:18:39.917       5330
## 9       163242                  The Pointer 2017-05-30T00:13:41.380       1344
## 10       56211                         rnso 2014-09-22T08:35:18.697       9299
##    Location Questions
## 1                 349
## 2                 298
## 3                 264
## 4    Europe       255
## 5                  236
## 6                  192
## 7                  184
## 8                  180
## 9                  166
## 10                 164
```

12. For each of the following terms, how many questions contain that term: Regression, ANOVA, Data Mining, Machine Learning, Deep Learning, Neural Network. For this, I interpret it as the title of a question containing any of these terms, not the body. So we will use the Posts table and filter it with LIKE each of the terms, grouping by so using CASE. (source: https://stackoverflow.com/questions/6101404/sql-group-by-like)

```
q = "SELECT Title, COUNT(*)
FROM Posts
WHERE (Title LIKE '%Regression%') OR (Title LIKE '%ANOVA%') OR (Title LIKE '%Data
↪  Mining%') OR (Title LIKE '%Machine Learning%') OR (Title LIKE '%Deep Learning%') OR
↪  (Title LIKE '%Neural Network%')
GROUP BY
  CASE
    WHEN Title LIKE '%Regression%' THEN 'Regression'
    WHEN Title LIKE '%ANOVA%' THEN 'ANOVA'
    WHEN Title LIKE '%Data Mining%' THEN 'Deep Mining'
    WHEN Title LIKE '%Machine Learning%' THEN 'Machine Learning'
    WHEN Title LIKE '%Deep Learning%' THEN 'Deep Learning'
    WHEN Title LIKE '%Neural Network%' THEN 'Neural Network'
    ELSE NULL
  END"

sql(q)
```

```
##                                                                                 Title
## 1                                                 How can I adapt ANOVA for binary data?
## 2                                    Deep learning vs. Decision trees and boosting methods
## 3                        Data Mining-- how to tell whether the pattern extracted is meaningful?
## 4                                         The Two Cultures: statistics vs. machine learning?
## 5            How to choose the number of hidden layers and nodes in a feedforward neural network?
## 6 Can one use multiple regression to predict one principal component (PC) from several other PCs?
##   COUNT(*)
## 1     2935
## 2      412
## 3      116
## 4     1638
## 5     2544
## 6    21886
```

I am not sure if SQLite supports the renaming of GROUP BY like the example on SO showed, but this does count the number of posts regardless, just without renaming. The group can be determined by the keyword in the titles.

14. What is the date range for the questions and answers in this database? To do this, we will look at the posts table and convert the creation date to a date and then find the minimum and maximum values for this. Then we will use julianday to do the difference between the two days, but since this is a lot of days, we will divide it 365 to get the number of years difference.

```
q = "SELECT MIN(datetime(CreationDate)), MAX(datetime(CreationDate)),
 ↪  (MAX(julianday(CreationDate)) - MIN(julianday(CreationDate))) / 365
FROM Posts"

sql(q)
```

```
##   MIN(datetime(CreationDate)) MAX(datetime(CreationDate))
## 1         2009-02-02 14:21:12         2023-03-05 05:10:18
##   (MAX(julianday(CreationDate)) - MIN(julianday(CreationDate))) / 365
## 1                                                             14.0921
```

15. What question has the most comments associated with it? • how many answers are there for this question? To do this, we will use the posts table and find the post with MAX(CommentCount). From this, we will also show the PostId, Title, and the answer count for this post. We will have to filter for questions again, so we will filter PostTypeId = 1 as known by the earlier command.

```
q = "SELECT Id, Title, MAX(CommentCount), AnswerCount, CreationDate
FROM Posts
WHERE PostTypeId = 1"

sql(q)
```

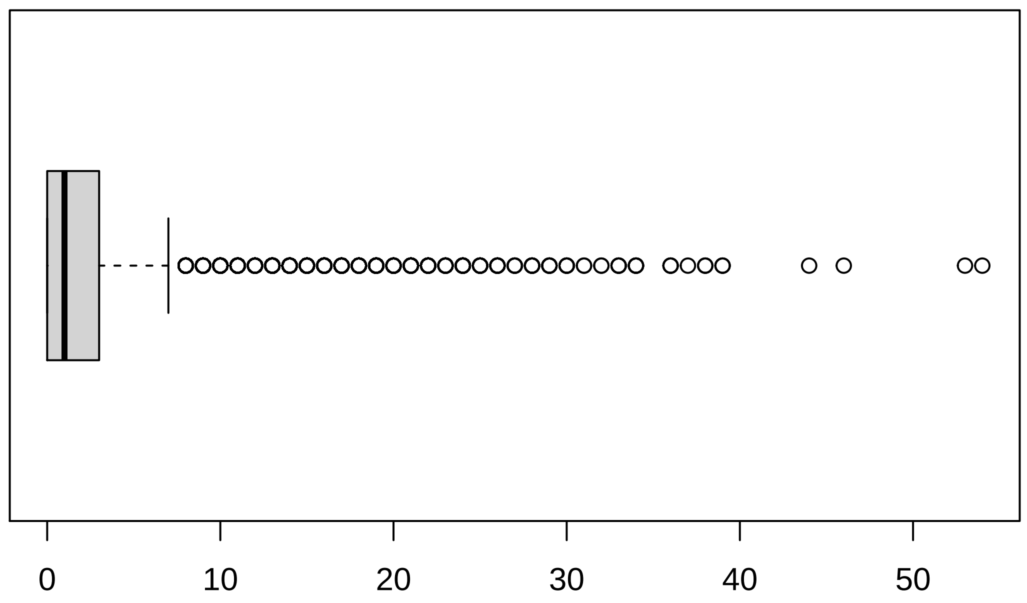
```
##       Id
## 1 328630
##                                                                             Title
## 1 Is ridge regression useless in high dimensions ($n \\ll p$)? How can OLS fail to overfit?
##   MAX(CommentCount) AnswerCount            CreationDate
## 1                54           6 2018-02-14T16:31:47.080
```

16. How many comments are there across all posts? • How many posts have a comment? • What is the distribution of comments per question? To do this, we will once again use only the Posts table. I do think we can answer the first two questions in one command, but the third question will need its own command so we can plot it. We will filter the posts such that each post in the table has at least one comment, then we will collapse the table by SUM() and COUNT(*). For the third question, we will filter only question by posttypeid and then we will plot a boxplot of the comment counts.

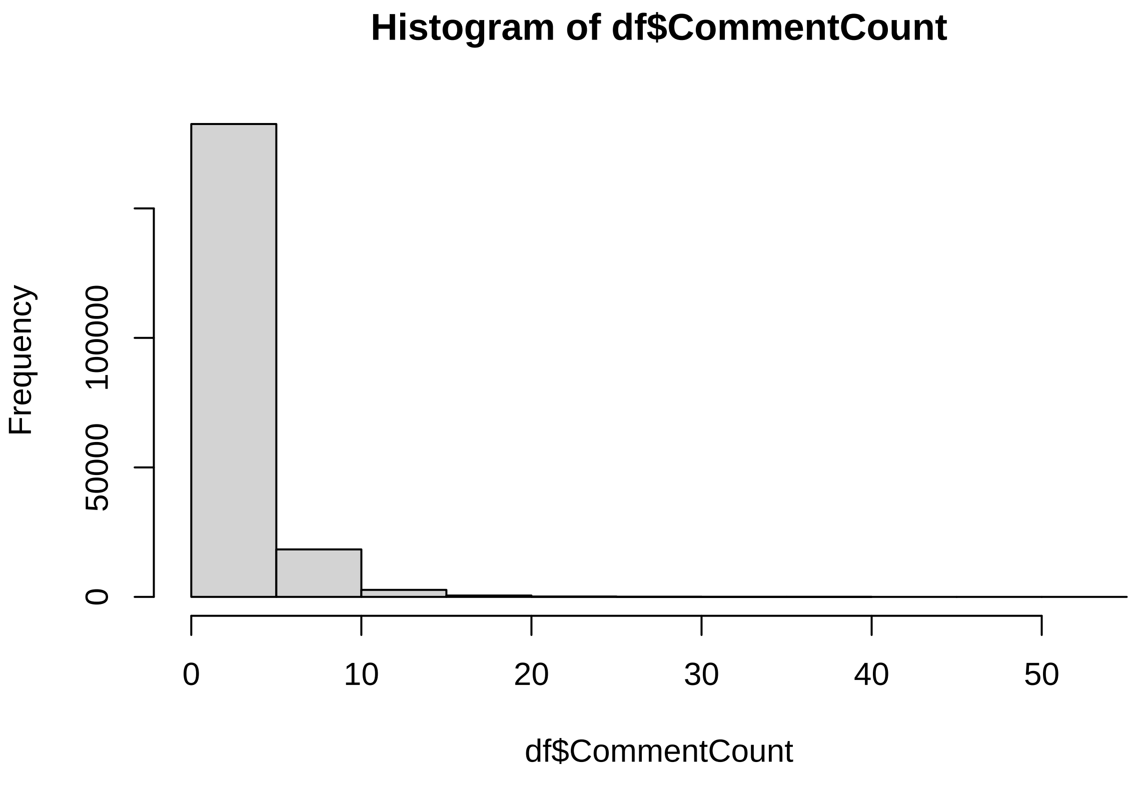
```
q = "SELECT SUM(CommentCount) as TotalComments, COUNT(*) as NumPostsWithComments
FROM Posts
WHERE CommentCount >= 1"

sql(q)
```

```
##   TotalComments NumPostsWithComments
## 1        768069               229859
```

```
q = "SELECT CommentCount
FROM Posts
WHERE PostTypeId = 1"

df = sql(q)

boxplot(df$CommentCount, horizontal = T)
```



```
hist(df$CommentCount)
```



```
summary(df$CommentCount)
```

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

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.

To do this, we will have to merge the Posts table with the users table, twice. Once to get the user with the question, and a second time to get the user with the answer so that we can join them both together. I simply went down the list to get each column needed: title of question, display name of question poster, creation date of user, location of user, date of question creation, the body of the accepted answer since these dont have titles, the date of the answer post, and the user who posted it.

```
q = "SELECT q.Title, u1.DisplayName, u1.CreationDate, u1.Location, q.CreationDate,
 ↪  a.Body, a.CreationDate, u2.DisplayName
FROM Posts as q, Posts as a, Users as u1, Users as u2
WHERE q.OwnerUserId = u1.Id
AND a.OwnerUserId = u2.Id
AND q.AcceptedAnswerId = a.Id
AND q.PostTypeId = 1"

head(sql(q))
```

```
##                                                                           Title
## 1                                                 Eliciting priors from experts
## 2                                                            What is normality?
## 3               What are some valuable Statistical Analysis open source projects?
## 4                            Assessing the significance of differences in distributions
## 5                                            Locating freely available data samples
## 6 Under what conditions should Likert scales be used as ordinal or interval data?
##   DisplayName            CreationDate                    Location
## 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      EAMann 2010-07-19T19:11:57.393 Tualatin, OR, United States
## 6      A Lion 2010-07-19T19:09:32.157
##              CreationDate
## 1 2010-07-19T19:12:12.510
## 2 2010-07-19T19:12:57.157
## 3 2010-07-19T19:13:28.577
## 4 2010-07-19T19:13:31.617
## 5 2010-07-19T19:15:59.303
## 6 2010-07-19T19:17:47.537
##
## 1
## 2
## 3
## 4
Smirnov test</a>, or the like.  The two-sample Kolmogorov-Smirnov test is based on comparing difference
## 5
## 6 <p>Maybe too late but I add my answer anyway...</p>\\\\n\\\\n<p>It depends on what you intend to d
##              CreationDate    DisplayName
## 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 2010-07-19T19:24:18.580 Stephen Turner
## 6 2010-08-19T10:00:00.370            chl
```

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? To do this, I will have to join the Posts and Users table once again. We will join on OwnerUserId = Id, then we will have to find the frequency of posts and answers. To do this, I found the most efficient way is to use SUM(CASE…) where we will have a case when PostTypeID = 1 and 2, so that the sum will be the frequency (source: https://stackoverflow.com/questions/24767130/get-frequency-of-a-column-in-sql-server). Then we can order by the number of answers to see those who have not answered. Then, to find those who have posted at least 1 question, I nested that query and subsetted it from there. I found it easiest to just table the number of answers to find the number of those who have not answered.

```
q = "SELECT Questions, Answers
FROM (

SELECT u.Id, u.DisplayName, SUM(CASE WHEN PostTypeId = 1 then 1 else 0 end) as Questions,
 ↪  SUM(CASE WHEN PostTypeId = 2 then 1 else 0 end) as Answers
FROM Posts as p, Users as u
WHERE p.OwnerUserid = u.Id
GROUP BY u.Id
ORDER BY Answers) as innerquery
WHERE Questions > 0"

df = sql(q)
head(df)
```

```
##   Questions Answers
## 1         2       0
## 2         2       0
## 3         1       0
## 4        13       0
## 5         2       0
## 6         1       0
```

```
head(table(df$Answers))
```

```
##
##     0     1     2     3     4     5
## 76410  6470  1864   888   557   372
```

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) Due to limitations of my machine, I could not get a decent runtime while using GROUP_CONCAT(b.Name) to put all badge names into one string, but this did work when I used LIMIT = 1, just to experiment, even then, it made R studio unbearable.

To do this question, it required a total of 4 tables, all left joins since this made it much simpler and quicker to run. I used Id from the Users table as the primary key that linked all the tables together and I grouped them on this Id. Users table contained all the information on the user that we needed. However, first we needed to calculate the top 75 accepted answers users, so we linked the posts table twice, one to the question and one to the accepted answer that linked back to the users table. From that, we could simply COUNT the distinct Id from the answers table so we could order by DESC. Then we included the badges table where we simply linked the Id's together and could COUNT Id from the badges table. As prefaced above, group_concat for the badge names would not run within a reasonable amount of time, but I did manage to group_concat for the tags since they were a shorter string. Min and Max datetime() were added too.

```
q = "SELECT u.Id, u.DisplayName, u.CreationDate, COUNT(DISTINCT a.Id) as AcceptedAnswers,
 ↪  COUNT(DISTINCT b.Id) as NumBadges, MIN(datetime(a.CreationDate)) as FirstAnswer,
 ↪  MAX(datetime(a.CreationDate)) as LatestAnswer, GROUP_CONCAT(DISTINCT q.Tags) as
 ↪  QuestionTags
FROM Users as u
LEFT JOIN Posts as q
ON q.OwnerUserID = u.Id
LEFT JOIN Posts as a
ON a.Id = q.AcceptedAnswerId
LEFT JOIN Badges as b
ON b.UserId = u.Id
GROUP BY u.Id
ORDER BY AcceptedAnswers DESC
LIMIT 75"

df = sql(q)
head(df)
```

```
##       Id                  DisplayName            CreationDate AcceptedAnswers
## 1   9162 user1205901 -         2012-02-13T02:09:08.377             179
## 2 108150                    user321627 2016-03-10T14:45:28.010             173
## 3 163242                   The Pointer 2017-05-30T00:13:41.380             124
## 4    253                    Tal Galili 2010-07-21T07:53:50.990             115
## 5  56211                          rnso 2014-09-22T08:35:18.697             112
## 6   1005                           Tim 2010-08-19T15:31:09.537             110
##   NumBadges         FirstAnswer        LatestAnswer
## 1       275 2012-05-16 05:36:25 2023-02-19 16:13:21
## 2        87 2016-03-10 15:11:46 2023-02-13 03:29:21
## 3        66 2017-06-06 10:41:22 2023-01-20 18:29:46
## 4       369 2010-08-09 18:00:24 2021-11-15 15:50:42
## 5       186 2014-10-01 09:50:21 2022-02-21 12:02:21
## 6       353 2010-09-30 16:32:06 2014-12-08 23:54:00
##
## 1        <multilevel-analysis><dyadic-data>,<forecasting><error><mse><mae>,<forecasting>,<confidence-
## 2 <mathematical-statistics><asymptotics><umvue>,<confidence-interval>,<mathematical-statistics><suffi
## 3
## 4
## 5
## 6
```

24. How many questions received no answers (accepted or unaccepted)? How many questions had no accepted answer? To do this in a simpler way, I broke it into two sql commands, one for the first question and one for the second. I used the same format for both, but on the first command, I linked

the two Posts tables with the parentId of the answer with the question Id. Then, since this was a left join, I could sum over the NULL values if the question did not have an answer. Same process for the accepted answer except that I linked the accepted answer id from the question to the id of the answer post.

```
q = "SELECT SUM(CASE WHEN a.id IS NULL THEN 1 ELSE 0 END) as NoAnswer
FROM Posts as q
LEFT JOIN Posts as a
ON a.ParentId = q.Id
WHERE q.PostTypeId = 1"

sql(q)
```

```
##   NoAnswer
## 1    66970
```

No Accepted Answers:

```
q = "SELECT SUM(CASE WHEN a.id IS NULL THEN 1 ELSE 0 END) as NoAcceptedAnswer
FROM Posts as q
LEFT JOIN Posts as a
ON q.AcceptedAnswerId = a.Id
WHERE q.PostTypeId = 1"

sql(q)
```

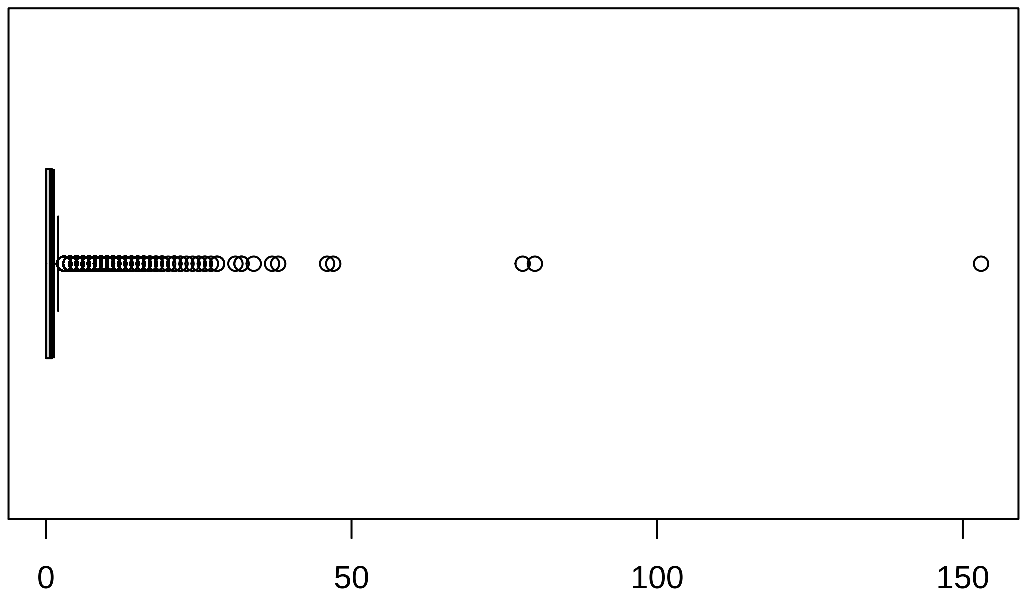
```
##   NoAcceptedAnswer
## 1           136366
```

25. What is the distribution of answers per posted question? For this, I tabled the answercount from the Posts table. From this I compared it to when I left joined Posts and Posts together on Id and ParentId. When I summed the number of answers, I could only get it to sum when it had one or more, ie it would not return 0 if a post did not have any answers. When comparing to the table of the first command, the numbers are the same except the first table has a count for 0 answers and the second does not. The distribution is very heavily weighted towards 0-1-2 answers and very few greater than that.

```
q = "SELECT q.Id, q.AnswerCount
FROM Posts as q
WHERE q.PostTypeId = 1"

df = sql(q)
table(df$AnswerCount)
```

```
## 
##     0     1     2     3     4     5     6     7     8     9    10    11    12 
## 66970 98602 27191  7246  2408   905   401   210   136    82    62    35    25 
##    13    14    15    16    17    18    19    20    21    22    23    24    25 
##    19    14    14     8     5     4     5     1     5     2     1     1     2 
##    26    27    28    31    32    34    37    38    46    47    78    80   153 
##     2     1     2     1     2     1     1     1     1     1     1     1     1 
```

```r
boxplot(df$AnswerCount, horizontal = T)
```



```r
summary(df$AnswerCount)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##   0.0000   0.0000   1.0000   0.9685   1.0000 153.0000
```

```r
q = "SELECT q.Id, SUM(CASE WHEN a.Id IS NOT NULL THEN 1 ELSE 0 END)
FROM Posts as q
LEFT JOIN Posts as a
ON a.ParentId = q.Id
WHERE a.PostTypeId = 2
AND q.PostTypeId = 1
GROUP BY q.Id"

df  = sql(q)

table(df$`SUM(CASE WHEN a.Id IS NOT NULL THEN 1 ELSE 0 END)`)
```

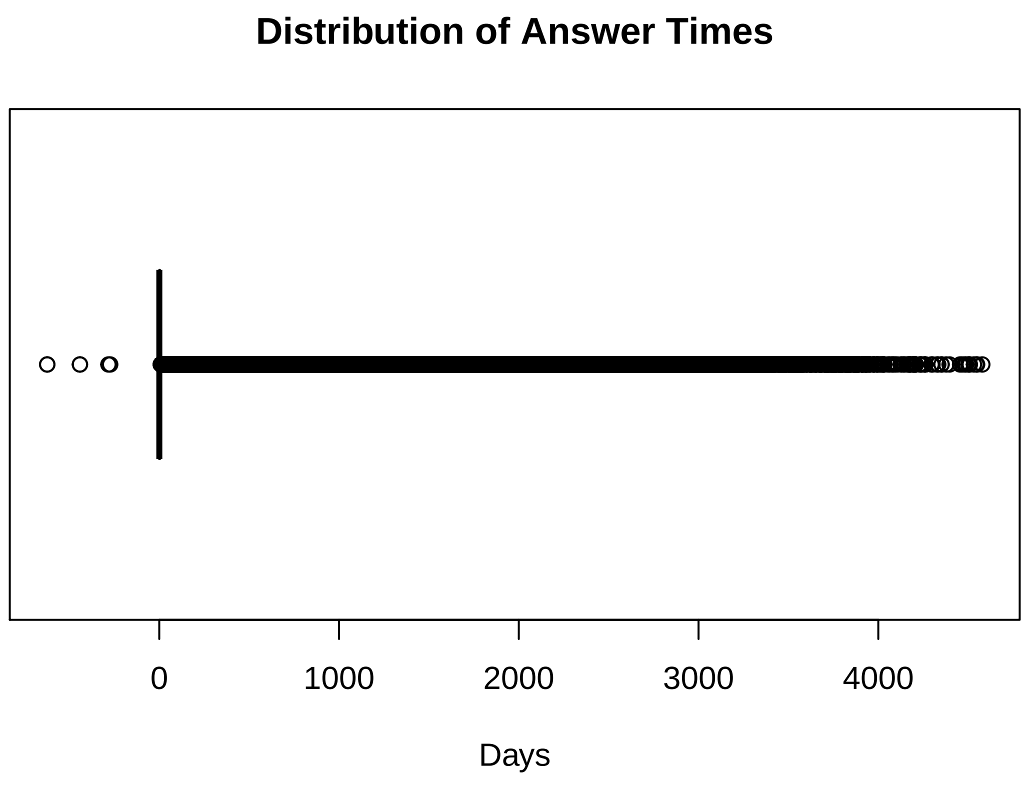
```
##
##     1     2     3     4     5     6     7     8     9    10    11    12    13
## 98602 27191  7246  2408   905   401   210   136    82    62    35    25    19
##    14    15    16    17    18    19    20    21    22    23    24    25    26
##    14    14     8     5     4     5     1     5     2     1     1     2     2
##    27    28    31    32    34    37    38    46    47    78    80   153
##     1     2     1     2     1     1     1     1     1     1     1     1
```

26. What is the length of time for a question to receive an answer? to obtaining an accepted answer? To do this, we will join the posts table with itself on Id = ParentId for all questions and Id = AcceptedAnswerId for the accepted answers. When this is done, we can see that there are some outliers where the answer is received before the question, but in most cases, the answer comes after the questions.

```
q = "SELECT q.Id, q.Title, q.CreationDate as QuestionDate, a.CreationDate as AnswerDate,
↪  julianday(a.CreationDate) - julianday(q.CreationDate) as AnswerTime
FROM Posts as q, Posts as a
WHERE q.Id = a.ParentId
AND q.PostTypeId = 1
AND a.PostTypeId = 2
ORDER BY AnswerTime"

df = sql(q)
head(df)
```

```
##       Id                                                            Title
## 1 112451                    Maximum Likelihood Estimation (MLE) in layman terms
## 2 112451                    Maximum Likelihood Estimation (MLE) in layman terms
## 3  16921                                   How to understand degrees of freedom?
## 4  16921                                   How to understand degrees of freedom?
## 5 116804 Coefficient changes sign when adding a variable in logistic regression
## 6 239431                                    Book recommendations for probability
##              QuestionDate              AnswerDate AnswerTime
## 1 2014-08-19T12:46:29.950 2012-12-03T23:52:18.377  -623.5376
## 2 2014-08-19T12:46:29.950 2012-12-04T13:09:31.343  -622.9840
## 3 2011-10-12T20:16:52.280 2010-07-28T12:48:14.233  -441.3116
## 4 2011-10-12T20:16:52.280 2010-07-28T12:49:31.780  -441.3107
## 5 2014-09-25T21:49:50.467 2013-12-16T21:13:01.827  -283.0256
## 6 2016-10-10T15:24:14.047 2016-01-11T15:19:13.797  -273.0035
```

```
boxplot(df$AnswerTime, horizontal = T, xlab = "Days", main = "Distribution of Answer
↪  Times")
```



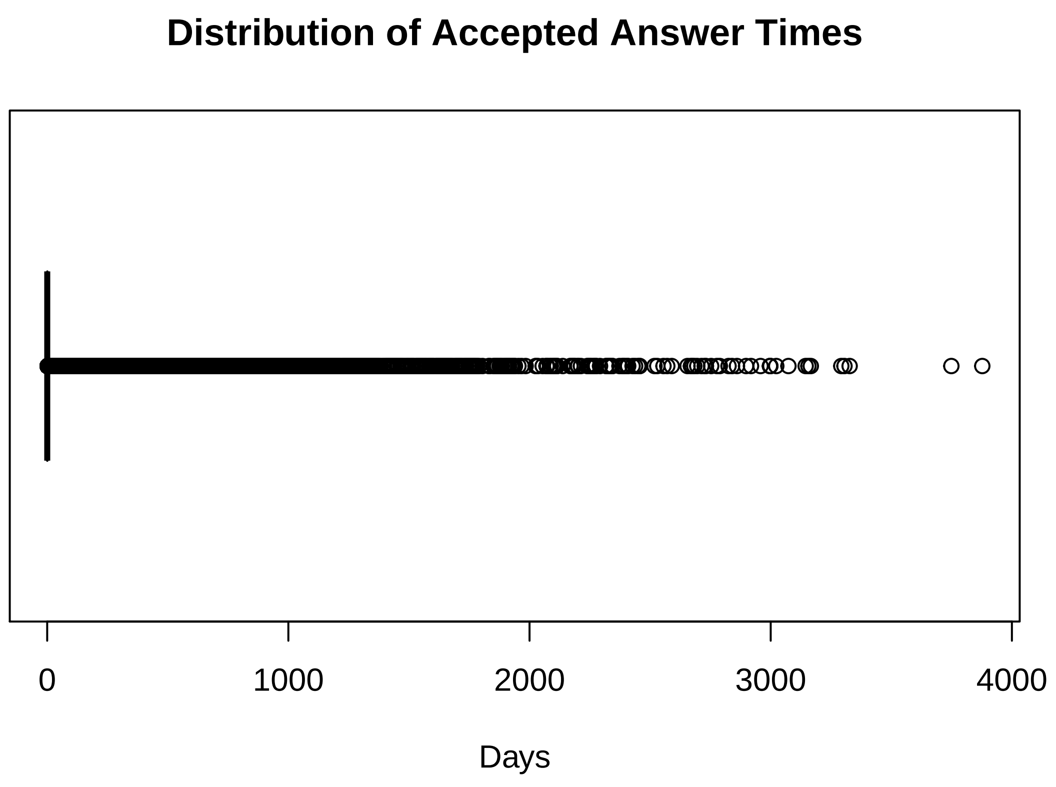
```
summary(df$AnswerTime)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -623.538    0.047    0.247  145.172    3.857 4578.530
```

```
q = "SELECT q.Id, q.Title, q.CreationDate as QuestionDate, a.CreationDate as AnswerDate,
↪  julianday(a.CreationDate) - julianday(q.CreationDate) as AnswerTime
FROM Posts as q, Posts as a
WHERE q.AcceptedAnswerId = a.Id
AND q.PostTypeId = 1
AND a.PostTypeId = 2
ORDER BY AnswerTime"

df = sql(q)
head(df)
```

```
##       Id                                                                 Title
## 1 129091                                Appropriate statistical analysis to use
## 2 108295                        Algorithm for rating books: Relative perception
## 3  34165              Expected value of the log-determinant of a Wishart matrix
## 4  51070    How can I convert a q-value distribution to a p-value distribution?
## 5  63391   Free internet or downloadable resources for sample size calculations
## 6  64538 How do I find values not given in (interpolate in) statistical tables?
##              QuestionDate              AnswerDate  AnswerTime
## 1 2014-12-14T20:27:18.950 2014-12-14T18:41:59.110 -0.07314630
## 2 2014-07-17T11:36:45.187 2014-07-17T11:09:39.077 -0.01882072
## 3 2012-08-12T03:36:55.910 2012-08-12T03:36:55.910  0.00000000
## 4 2013-02-28T20:05:01.313 2013-02-28T20:05:01.313  0.00000000
## 5 2013-07-05T04:10:52.657 2013-07-05T04:10:52.657  0.00000000
## 6 2013-07-16T23:55:34.533 2013-07-16T23:55:34.533  0.00000000
```

```
boxplot(df$AnswerTime, horizontal = T, xlab = "Days", main = "Distribution of Accepted
↪  Answer Times")
```



```
summary(df$AnswerTime)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##   -0.073    0.036    0.141   25.306    0.919 3877.244
```

27. How many answers are typically received before the accepted answer? To do this, I first did this individually for each question, then found the average in another command after I knew this would work. I joined Posts three times together, once for the question, second time for all the answers, and third time for the accepted answer. From this, I used SUM(CASE) once again such that it summed all the times when the creation date of the answer not accepted was less than the creation date of the accepted answer. From there, I averaged all of the values in another sql command by using the first command as a nested query. I did mean() in R to validate the results.

```
q = "SELECT q.Id, q.Title, SUM(CASE WHEN julianday(a.CreationDate) <
 ↪  julianday(ac.CreationDate) THEN 1 ELSE 0 END) as NumAnswersBeforeAccepted
FROM Posts as q, Posts as a, Posts as ac
WHERE q.Id = a.ParentId
AND q.AcceptedAnswerId = ac.Id
AND q.PostTypeId = 1
AND a.PostTypeId = 2
AND ac.PostTypeId = 2
GROUP BY q.Id"
df = sql(q)
head(df)
```

```
##   Id
## 1  1
## 2  2
## 3  3
## 4  4
## 5  7
## 6 10
##                                                                         Title
## 1                                                Eliciting priors from experts
## 2                                                           What is normality?
## 3                What are some valuable Statistical Analysis open source projects?
## 4                       Assessing the significance of differences in distributions
## 5                                          Locating freely available data samples
## 6 Under what conditions should Likert scales be used as ordinal or interval data?
##   NumAnswersBeforeAccepted
## 1                        0
## 2                        1
## 3                        0
## 4                        1
## 5                        1
## 6                        2
```

```
mean(df$NumAnswersBeforeAccepted)
```

```
## [1] 0.1836068
```

```
q = "SELECT AVG(NumAnswersBeforeAccepted)
FROM
(SELECT q.Id, q.Title, SUM(CASE WHEN julianday(a.CreationDate) <
 ↪  julianday(ac.CreationDate) THEN 1 ELSE 0 END) as NumAnswersBeforeAccepted
FROM Posts as q, Posts as a, Posts as ac
WHERE q.Id = a.ParentId
AND q.AcceptedAnswerId = ac.Id
AND q.PostTypeId = 1
AND a.PostTypeId = 2
AND ac.PostTypeId = 2
GROUP BY q.Id) as innerquery"

sql(q)
```

```
##   AVG(NumAnswersBeforeAccepted)
## 1                     0.1836068
```