Assignment 1

Due date: January 28, 2020, 11:55pm IST

General Instructions

1. Please complete this assignment *individually*, on your own.
2. You will submit 2 files: query.sql and *EntryNumber*.pdf, corresponding to the queries and a timings report, respectively.
3. Use PostgreSQL 12 for your homework. See **this link** for instructions on how to download and install it on your OS. The .sql files are run automatically using the psql command using the \i option, so please ensure that there are no syntax errors in the file. *If we are unable to run your file, you get an automatic reduction to 0 marks.*

   To understand how to run many queries at once from text file, a dummy query file **example.sql** is available. To run **example.sql** in PostgreSQL, type the following command in the terminal:

   sudo -u postgres psql *dbname*

   `\i /address/to/example.sql`

   This command will run all the queries listed in example.sql at once.
4. The format of the file should be as follows. You can have a preamble where you may create views if you like (please note that no procedures are allowed (this has to be pure SQL), and correspondingly have a cleanup section where these views are removed. One line should identify the query number (note the two hyphens before and after the query number), followed by the actual, syntactically correct SQL query. Leave a blank line after each query.

   - -PREAMBLE- -

   OPTIONAL DEFINITIONS

   - -1- -

   SQL QUERY

   - -2- -

   SQL QUERY

   - -3- -

   SQL QUERY

   - -CLEANUP- -

   CLEANUP EVERYTHING YOU CREATED HERE
5. All of the queries below require an 'ORDER BY' clause. If you made an error in this clause, your answer will be evaluated as incorrect and zero marks will be awarded.
6. No changes are allowed in the i) data, ii) attribute names, iii) table names
7. The .pdf file should contain a bar graph. The graph should report the timings for each query of the dataset (X-axis legend is the query number, Y-axis legend is the time taken). You will need to figure out how to measure the timings.
8. The submission will be done on Moodle.
9. If unspecified, order in ascending order by column 1, then column 2 .. etc. In case of any doubts please ask on Piazza. The instructors ordering will be final and no queries will be entertained on the same.

10. In case any query or sub-query results in NULL values, please discard them before proceeding for any further steps. This is extremely important for some queries which ask us to list TOP n results etc, since you might get erroneous results.

11. For all queries leading to floating point numbers, round of the number to two decimal places right when the number is computed (and not just at the end). Even a 0.01 error will result in a wrong answer.

12. For all queries regarding dates or month, day, year etc. please use the full date to compute the duration or time-period etc. For example the time may determine if a certain date falls within a 6-month period or not, so be sure to handle such corner cases

1 Dataset

1.1 Instructions

1. In this assignment you will analyze stackoverflow data from the years 2008-2010, the reference citation website created and maintained by Stack Exchange. The analysis has to be done in postgres. We are providing you cleaned up data and you can download it from **this link**. The zip file contains a pipe-seperated (|) file for each table described below.(Note the order of values in file is same as attributes of table given in next bullet point). You can load the table into database from csv file using the command

   ```
   copy Table-Name from '/path/to/file/table-name.csv' DELIMITER '|' CSV HEADER;
   ```

2. Please refer to the *LinkTypes.csv* and *postTypes.csv* for the specific details on LinkTypeId and PostTypeId. Wherever asked for questions/answers, please use only the specific PostTypeId.

3. The database will include following eight tables and you should use only these tables while writing solution of the queries. All blue coloured values are the PostId and the red ones are UserId across all tables. You can create temporary views while handling any SQL query but you should include SQL queries for creating and deleting these temporary views at the starting and end of your SQL file respectively. Note - you don't have to define these tables in the submission file, these will already be present will evaluation.

   (a) Badges

Id : integer	Name : text	UserId : int	Date : timestamp

   (b) Comments

Id : integer	CreationDate : timestamp	PostId : integer
Score : integer	UserId : integer	

   (c) PostLinks

Id : integer	CreationDate : timestamp	PostId : integer
RelatedPostId : integer	LinkTypeId : integer	

   Note : Only links from postid to relatedpostid are considered. Do not assume two way linkage

   (d) Posts

Id : integer	AcceptedAnswerId : integer	AnswerCount : integer
ClosedDate : timestamp	CommentCount : integer	CommunityOwnedDate : timestamp
CreationDate : timestamp	FavoriteCount : integer	LastActivityDate : timestamp
LastEditDate : timestamp	LastEditorDisplayName : text	LastEditorUserId : integer
OwnerUserId : integer	ParentId : integer	PostTypeId : integer
Score : integer	Tags : text	Title : text
ViewCount : integer		

   (e) Users

Id : integer	Age : integer	CreationDate : timestamp
DisplayName : text	DownVotes : integer	EmailHash : text
LastAccessDate : timestamp	Location : text	Reputation : integer
UpVotes : integer	Views : integer	WebsiteUrl : text
AccountId : integer		

   (f) Votes

Id : integer	PostId : integer	UserId : integer
BountyAmount : integer	VoteTypeId : integer	CreationDate : timestamp

1.2 Queries

1. List all the names and count of badges that have be given to atleast 10,000 people, sorted in descending order of their counts. Break ties by ascending order of badge name. **Columns: name, number**

2. List the top 5 users (userid and displayname) with the maximum number of comments across all posts. Resolve ties by alphabetic ordering of username. **Columns: userid, displayname**

3. List users that earned maximum badges in each month of 2010. Break ties using lexicographic ordering of user names. There should be exactly **12** rows in output, one for each month. Order by ascending month. (Here month will be a number between 1-12) **Columns: month,userid.**

4. List post ids whose title contain more than 100 characters. Output (post id and number of characters) in decreasing order of this number, while breaking ties using post ids in ascending order. **Columns:** postid, charcount

5. List the first 5 column names of the posts table that occur in ascending order. **Columns: postcolumn**

6. Give the title of the top 5 posts which have the maximum number of linked posts (Assume only related post Id and not Id). Break ties by alphabetic ordering of titles. Order by descending order of number of links. **Columns: title, count**

7. Which user names have created more than one post in any 24-hour interval? List user names in lexicographic order. Here, 24-hour interval means the timestamp value differs by not more than 24-hours **Columns: displayname**

8. List top 3 user names that have offered the most bounty amount (concerns only with bounty start and not bounty close). Break ties using lexicographic ordering of user names. **Columns: displayname**

9. List the top 5 user names who have made the maximum number of last-edits on a post and have atleast 10 badges. Break ties by alphabetical ordering of names. **Columns: displayname**

10. List the user for every month of every year with maximum absolute difference between questions asked and questions answered in that month. Break ties with lexicographic ordering of user name. Output (user id) for each month. **Columns: year, month, userid**

11. List the average view count for questions per tag. Display the tag name and the average view count. Sort by tag name in ascending alphabetic order. **Columns: tags, viewcount**

12. List the top 10 tags with most unanswered questions. Break ties with tag-post count and if still ties exist, use lexicographic ordering of tag name. **Columns: tags, number**

13. List users (user names) that wrote more than one answer for a single post within 24 hours (Creation-Date) (time of last answer - first answer $<=$ 24 hours), with one of the answers being an accepted answer. **Columns: displayname**

14. List the Top 10 users, who have earned maximum badges, sorted in descending order of total badges. **Columns: userid, totalbadges.**

15. Which day of the week has the most unanswered questions with atleast 10 views posted on it? **Columns: day -> text**

16. Which posts have the highest votes (Upvote or downvote) to views ratio (With atleast 1 view)? List the top 10, along with the ratio. Order by decreasing ratio. **Columns: postid, ratio**. Here ratio will be a floating point number, so please round to 2 decimal places at all places of calculation.

17. Give the username of the top 3 people who have received the max total comments score on a single post across all comments made by them. Break ties in ascending alphabetical order of names. **Columns: displayname**

18. Which old users (user account was created in 2008 and atleast 10 accepted answers) are not active any more (no login in 2010)? List (user name, number of accepted answers) in decreasing order of number of accepted answers. Break ties with lexicographic ordering of user names. **Columns: displayname, number**

19. List the post (postid) that has maximum effective upvotes but is still unanswered. Effective upvotes = Total upvotes - Total downvotes. **Columns: postid, effectiveupvotes** Hint: Upvotes correspond to voteTypeId of 2 and Downvotes correspond to a voteTypeId of 3. Please ignore all other voteTypeIds.

20. List users that have at least 100 badges such that they have at least one post with a view count higher than their profile view count (post may be question or answer only). Output in decreasing order of the difference of views. Break ties with lexicographic ordering of user name. **Columns: userid, viewdiff**

21. Find the number of average replies per question by expert askers >1000 reputation) and also the average of new askers (<100 reputation) **Columns: askertype, replies.** Here askertype can be expert or new. The output will have two rows, one for expert askers and one for new askers.

22. Find the number of lurkers (ie number of users who have never asked or answered or replied) who have had an account for atleast 6 months. **Columns: lurkercount.**

1.3 Additional Resources

1. Can check out more about the dataset at Google BigQuery
2. This smaller dataset was created by Brent Ozar
3. Further details about the schema here
4. In case any student wants to explore further the full dataset can be downloaded from here