askreddit

Mira RAHAL (21911563)

Mohar CHAUDHURI (21911249)

Supervised by : Ronan TOURNIER

Toulouse School of Economics

M2 Statistics and Econometrics

April 6, 2021

Table of Contents

1 Introduction

Our study aims at analysing the activity and participation in the rising social news website, Reddit. It is driven by user content and the content is either a link to another webpage or a text blog accompanied by a title. Each submission is voted upon by users of the site, either up or down, to rank the submission. Users can also discuss the submission in a threaded comment section attached to each submission. There are close to 25,000 subreddits with many of them empty or non-active. We are using the reduced dataset, the comments of the subreddit "AskReddit". The time period of study is from May 01, 2015 to May 30, 2015.

The data under study can be used to assemble a database described by the following entity/relationship diagram.

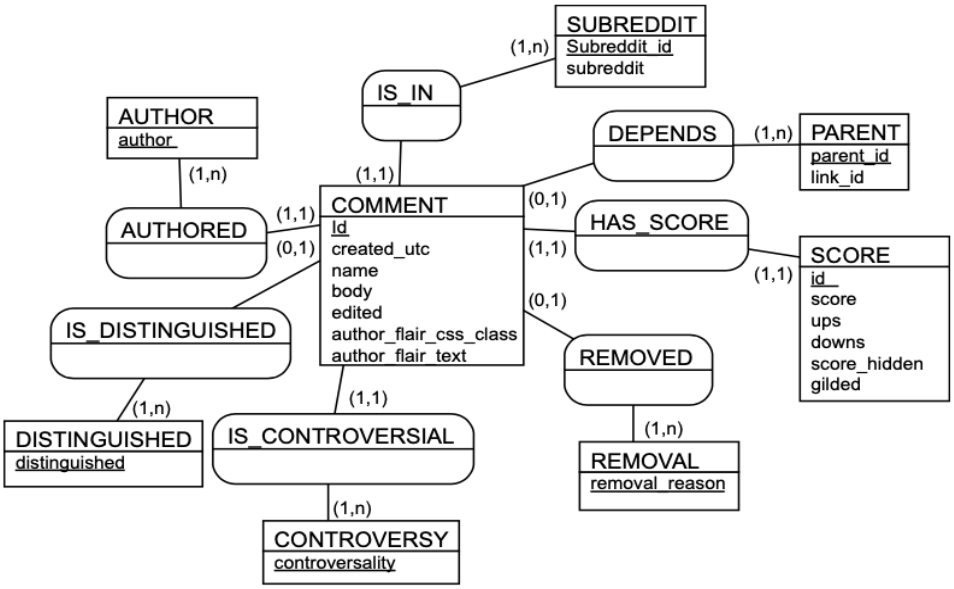

Figure 1: Entity/Relationship diagram of the database

This database is translated into the following relational schema where primary keys are underlined and foreign keys are followed by the # sign.

- **AUTHOR** (author)
- **DISTINGUISHED** (distinguished)
- **CONTROVERSY** (controversiality)
- **REMOVAL** (removal_reason)
- **SCORE** (id#,score, ups, downs, score_hidden, gilded)
- **PARENT** (parent_id, link_id)
- **SUBREDDIT** (subreddit_id, subreddit)
- **COMMENT** (,created_utc, name, body, edited, author_flair_css_class, author_flair_text, author#, controversiality#, subreddit_id#)
- **IS_DISTINGUISHED** (id#, distinguished#)
- **REMOVED** (id#, removal_reason#)
- **DEPENDS** (id#, parent_id#

2 Creating and populating the database

We have been provided with some *.csv* files and a data dictionary describing the meaning of each of the variables in the files.

- *exp_author.csv*
- *exp_comment.csv*
- *exp_controversy.csv*
- *exp_distinguihshed.csv*
- *exp_parent.csv*
- *exp_removal.csv*
- *exp_score.csv*
- *exp_subreddit.csv*

The first step of this project was to insert data in SQLite in order to be able to query and analyse the data. Since we have *.csv* files at disposal, we attempted to insert the data into SQLite from these files.

However, if we use SQLite to create a table from a csv file automatically, all the columns will be of the TEXT data type and will thus have to be changed into the correct datatype. Hence, to avoid this complication we employ the following steps :

1. Create all the required database tables.
2. Load the *.csv* in a temporary table.
3. Use the insert data from a query statement to populate the database tables from the temporary table
4. Delete the temporary tables after all the data has been inserted.

Now, we will explain each and every SQL query to create the database with the corresponding tables.

2.1 Creating an empty database

```
-- Creating the Database --
sqlite3 askreddit_data.db
--.open askreddit_data.db -- to access the database later
```

At a shell or Terminal, we enter the above queries and this creates a new database named *"askreddit_data.db"*. Following this, we check if the database already has the tables we will be creating. If so, we delete them.

```
-- Checking if these tables already exist --
DROP TABLE IF EXISTS author;
DROP TABLE IF EXISTS distinguished;
DROP TABLE IF EXISTS controversy;
DROP TABLE IF EXISTS removal;
DROP TABLE IF EXISTS score;
DROP TABLE IF EXISTS parent;
DROP TABLE IF EXISTS subreddit;
DROP TABLE IF EXISTS "comment";
DROP TABLE IF EXISTS is_distinguished;
DROP TABLE IF EXISTS removed;
DROP TABLE IF EXISTS depends;
```

After checking for existing tables, we proceed to create some empty tables which we will populate with the *.csv* files later. It is essential that we start with the tables that have no foreign keys. With the help of the relational schema described above, we start to create these empty tables.

```
CREATE TABLE author(
        author   TEXT,
        --CONSTRAINTS (key)
        CONSTRAINT pk_author PRIMARY KEY (author)
);
```

The primary key of the **author** table is *author*. We have used the TEXT type for this variable because *author* is the account name of the poster of the comment. It is in the string format.

```
CREATE TABLE distinguished(
        distinguished    TEXT,
```

```
        --CONSTRAINTS (key)
        CONSTRAINT pk_distinguished PRIMARY KEY (distinguished)
);
```

The primary key of the **distinguished** table is *distinguished*. We have used the TEXT type for this variable because *distinguished* is a string which tells us an author is distinguished or not. If he is distinguished, he can be either a moderator or a special.

```
CREATE TABLE controversy(
        controversiality          INT,
        --CONSTRAINTS (key)
        CONSTRAINT pk_controversy PRIMARY KEY (controversiality)
);
```

The primary key of the **controversy** table is *controversiality*. We have used the INT type for this variable because *controversiality* is a integer taking values 0 or 1.

```
CREATE TABLE removal(
        removal_reason    TEXT,
        --CONSTRAINTS (key)
        CONSTRAINT pk_removal PRIMARY KEY(removal_reason)
);
```

The primary key of the **removal** table is *removal_reason*. We have used the TEXT type for this variable because *removal_reason* is a string.

```
CREATE TABLE Parent(
        parent_id         TEXT,
        link_id       TEXT,
        --CONSTRAINTS (key)
        CONSTRAINT pk_parent PRIMARY KEY(parent_id)
);
```

The primary key of the **parent** table is *parent_id*. There is another column named *link_id*. We have used the TEXT type for both the variables because they are in string format. *link_id* is the ID of the link a comment is in. While, *parent_id* is the ID of the comment, this comment is a reply to.

```
CREATE TABLE subreddit(
        subreddit_id    TEXT,
        subreddit       TEXT,
        --CONSTRAINTS (key)
        CONSTRAINT pk_subreddit PRIMARY KEY(subreddit_id)
);
```

The primary key of the **subreddit** table is *subreddit_id*. There is another column named *subreddit*. We have used the TEXT type for both the variables because they are in string format. *subreddit_id* is the id of the subreddit in which the comment is located and *subreddit* is the subreddit of the comment excluding the /r/ prefix.

We now proceed to create the tables with foreign keys, i.e., tables which have relations with the above created tables.

```
CREATE TABLE "comment"(
        id              TEXT,
        created_utc     INT,
        name            TEXT,
        "body"          TEXT,
    edited          INT,
        author_fair_css_class   TEXT,
        author_fair_text        TEXT,
        author                  TEXT,
        controversiality        TEXT,
        subreddit_id            TEXT,
        --CONSTRAINTS (key)
        CONSTRAINT pk_comment PRIMARY KEY(id),
        CONSTRAINT fk_comment_author FOREIGN KEY (author)
        REFERENCES author (author),
        CONSTRAINT fk_comment_controversy FOREIGN KEY (controversiality)
        REFERENCES controversy (controversiality),
        CONSTRAINT fk_comment_subreddit FOREIGN KEY (subreddit_id)
        REFERENCES subreddit (subreddit_id)
);
```

We want to highlight on the fact that the name of the table is in quotations here. This is because 'comment' is a keyword in SQLite. Everytime we come across a word which is a keyword in SQLite, we will use quotation marks to distinguish it.

The primary key of the **"comment"** table is *id*. It is a comments's identifier, e.g. "8xwlg". AS it is in a string format, we use the TEXT type. There are some other columns namely, *created_utc*, *name*, *"body"*, *edited*, *author_fair_css_class* and *author_fair_text*.

- *created_utc* is the time of creation of the comment in UTC epoch-second format. It is a long integer and so we have used the INT type.
- *name* is the full-name of comment, e.g. "t1_c3v7f8u". It is a string so we have used the TEXT type.
- *"body"* contains the raw text of the comment. This is an unformatted text which includes the raw markup characters such as ** for bold. <, >, and & are escaped. It is a string so we have used the TEXT type.
- *edited* is 0 if the comment is not edited and edit date in UTC epoch-seconds otherwise. It is a long integer and so we have used the INT type.
- *author_fair_css_class* is the CSS class of the author's flair. It is subreddit specific and as it is a string so we have used the TEXT type.
- *author_fair_text* is the the text of the author's flair. It is again subreddit specific and as it is a string so we have used the TEXT type.

Now let us introduce the concept of foreign keys. **A FOREIGN KEY** is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table. We have 3 foreign keys in the **"comment"** table :

- *author* which is the primary key of the **author** table
- *controversiality* which is the primary key of the **controversy** table.
- *subreddit_id* which is the primary key of the **subreddit** table.

```
1 CREATE TABLE score(
2         id              TEXT,
3         score           INT,
4         ups             INT,
5         downs           INT,
6         score_hidden    BOOLEAN,
7         gilded          INT,
```

```
        --CONSTRAINTS (key)
        CONSTRAINT pk_score PRIMARY KEY(id),
        CONSTRAINT fk_score_comment FOREIGN KEY (id)
        REFERENCES "comment" (id)
);
```

The primary key of the **score** table is *id*. It is a also a foreign key, because it is the primary key of the **"comment"** table. There are some other columns namely, *score*, *ups*, *downs*, *score_hidden* and *gilded*.

- *score* is the the net-score of the comment. It is an integer and so we have used the INT type.
- *ups* is the number of up votes (including own) that the comment receives. It is an integer so we have used the INT type.
- *downs* is the number of down votes (including own) that the comment receives. It is an integer so we have used the INT type.
- *score_hidden* is a binary binary variable telling us if or not the score is hidden. We use the BOOLEAN type for this.
- *gilded* is the number of times this comment received reddit gold. It is an integer so we have used the INT type.

```
CREATE TABLE is_distinguished(
        id                  TEXT,
        distinguished       TEXT,
        --CONSTRAINTS (key)
        CONSTRAINT pk_is_distinguished PRIMARY KEY(id,distinguished),
        -- We have to specify that the foreign keys get updated when
        -- the table of reference is updated
        CONSTRAINT fk_is_distinguished_comment FOREIGN KEY (id)
        REFERENCES "comment" (id) ON UPDATE CASCADE,
        CONSTRAINT fk_is_distinguished_distinguished FOREIGN KEY
        (distinguished) REFERENCES distinguished (distinguished)
        ON UPDATE CASCADE
);
```

The primary keys of the **is_distinguished** table are *id* and *distinguished*. These are also the foreign keys, because they are the primary keys of the **"comment"** and **distinguished** tables respectively.

```
CREATE TABLE removed(
        id                  TEXT,
        removal_reason      TEXT,
        --CONSTRAINTS (key)
        CONSTRAINT pk_removed PRIMARY KEY(id,removal_reason),
        CONSTRAINT fk_removed_comment FOREIGN KEY (id)
        REFERENCES "comment" (id) ON UPDATE CASCADE,
        CONSTRAINT fk_removed_removal FOREIGN KEY (removal_reason)
        REFERENCES removal (removal_reason) ON UPDATE CASCADE
);
```

The primary keys of the **removed** table are *id* and *removal_reason*. These are also the foreign keys, because they are the primary keys of the **"comment"** and **removal** tables respectively.

```
CREATE TABLE depends(
        id          TEXT,
        parent_id     TEXT,
        --CONSTRAINTS (key)
        CONSTRAINT pk_depends PRIMARY KEY(id,parent_id),
        CONSTRAINT fk_depends_comment FOREIGN KEY (id)
        REFERENCES "comment" (id) ON UPDATE CASCADE,
        CONSTRAINT fk_depends_parent FOREIGN KEY (parent_id)
        REFERENCES parent (parent_id) ON UPDATE CASCADE
);
```

The primary keys of the **depends** table are *id* and *parent_id*. These are also the foreign keys, because they are the primary keys of the **"comment"** and **parent** tables respectively.

We can do a final check up to see whether all of our tables were created successfully

```
.tables
-- Success ! :)
```

2.2 Loading the data into the database

We explained in Section 2 that in order to populate the tables with data we will create temporary tables so that we don't get mixed up with the data types.

```
.mode csv
.import exp_author.csv author_temp
.import exp_distinguihshed.csv distinguished_temp
.import exp_controverse.csv controversy_temp
.import exp_removal.csv removal_temp
.import exp_parent.csv parent_temp
.import exp_subreddit.csv subreddit_temp
.import exp_comment.csv "comment_copy"
.import exp_score.csv score_temp
```

Now, we proceed to import all the data from the above created temporary tables into the original tables we had created initially in Section 2.1. We follow the order of creation of the original tables.

```
INSERT INTO author SELECT * FROM author_temp;
INSERT INTO distinguished SELECT * FROM distinguished_temp;
INSERT INTO controversy SELECT * FROM controversy_temp;
INSERT INTO removal SELECT * FROM removal_temp;
INSERT INTO parent SELECT * FROM parent_temp;
INSERT INTO subreddit SELECT * FROM subreddit_temp;
INSERT INTO "comment" SELECT * FROM "comment_copy";
INSERT INTO score SELECT * FROM score_temp;
```

After the insertion of the data we delete all the temporary tables.

```
DROP TABLE author_temp;
DROP TABLE distinguished_temp;
DROP TABLE controversy_temp;
DROP TABLE removal_temp;
DROP TABLE parent_temp;
DROP TABLE subreddit_temp;
DROP TABLE "comment_copy";
DROP TABLE score_temp;
```

We are left to insert data into the 3 remaining tables **is_distinguished**, **removed** and **depends**. These tables are those that correspond to all associations in the entity relationship diagram with a (0,1) cardinality.

To insert data into these tables we need to use the other file provided to us. We follow the similar steps as explained above.

```
-- Similarly, we create a temporary table

.import exp_askreddit.csv full_table_temp
INSERT INTO is_distinguished SELECT id, distinguished
FROM full_table_temp;
INSERT INTO removed SELECT id, removal_reason FROM full_table_temp;
INSERT INTO depends SELECT id, parent_id FROM full_table_temp;

-- Now we delete the temporary table
DROP TABLE full_table_temp;
```

We have successfully created the required tables and import data into them. In table 1 we show the output of from the server to show the how many rows have been inserted in each table.

Table Name	Number of Rows
author	570735
distinguished	3
controversy	2
removal	1
score	4234970
parent	1464558
"comment"	4234970
is_distinguished	4234970
removed	4234970
depends	4234970

Table 1: Row count for each table

Next, in the study we will try to explore the database using SQL. We shall divide the next section of exploratory data analysis into two sub parts, one for basic/general analyses and the other for more advanced/detailed analyses.

3 Exploring the database using SQL

3.1 Basic Analysis : What is inside the database?

As we have already seen in the previous section, we have a database with 11 tables. These are the Reddit comments for the subreddit, AskReddit. We will be mainly focusing on the comments and the attributes associated to it. So, let us note down some pointers :

- A comment is made by an author.
- The comment comes with an associated score, which is the net of upvotes and downvotes.
- The score of the comment can be hidden.
- We also have a count for the number of times a comment received reddit gold.
- The comment made may or may not be distinguished.
- It may or may not be controversial.
- It may or may not be edited.
- Again, the comment may or may not be removed.

Before we start from query, we want to introduce the concept of indexes in SQL. Access to data in databases can be highly accelerated using indexes. An **index** is an on-disk structure associated with a table that speeds retrieval of rows from the table. It contains keys built from one or more columns in the table. These keys are stored in a structure (B-tree) that enables SQL Server to find the row or rows associated with the key values quickly and efficiently. The syntax for indexing is the following :

```
CREATE INDEX index_name
ON table_name (column1, column2, ...);
```

We have created indexes when and where we felt necessary, the details of which are attached in the SQL code provided.

Now, we want to explore the basic attributes about the comments. We will for now only provide the count and percentage of comments with these attributes. Later, we will delve deeper into them.

Attribute	Count	Percentage(%)
Distinguished	39764	0.94
Controversial	52218	0.01
Edited	80620	1.90
Hidden Scores	7011	0.16
Received Gold Reddits	2991	0.07
Removed	0	0.00

Table 2: Statistics on attributes of comments

Table 2 has been populated with two similar types of queries.

```
-- How many comments were edited ?
SELECT COUNT(id) as number_edited
        FROM "comment" as c
        WHERE c.edited <> 0;
-- Percentage of comments that are edited
SELECT ((COUNT(id)*100.0) /(select count(*) from "comment"))
        FROM "comment" as c
                WHERE c.edited <> 0;
```

Similar queries were implemented for finding the counts and percentages of the other attributes mentioned in Table 2.

Our database is quite big with 4234970 comments. The attributes affect a very small number of comments, none above 2% of the total comments. This is an interesting point because in the study later, we will describe a sub selection of the data that would represent only about 5MB. For now, we continue with the basic analyses and find the summary statistics for the score of the comments.

According to the data dictionary, the score of the comment is basically the net-score of the comment,

$$\texttt{Score} = \texttt{Upvotes} - \texttt{Downvotes}$$

However, we notice that in the database we are working with, the score coincides with the upvotes. We also note that negative values of the columns *ups* can indicate down votes, as the *down* column in our database is always 0, which we discovered by running the following queries. So, we decide to ignore the *ups* and *down* columns, only focusing on the *score*.

```
SELECT MIN(downs) as min_downs FROM score;
SELECT MAX(downs) as max_downs FROM score;
```

Mean	Median	Mode	Min	Max	Standard Deviation[1]
12.60	1	1	-333	6761	12.56

Table 3: Summary Statistics on *Score*

The SQL code for the summary statistics computed in Table 3 are the following:

```
-- Mean
SELECT AVG(score) as average_score FROM score;
-- Median
SELECT score.score  as median_score
FROM score
ORDER BY score.score
LIMIT 1
OFFSET (SELECT COUNT(*) FROM score) / 2
-- Mode
SELECT  score.score, COUNT(*) as modecount
FROM score
GROUP BY score.score
ORDER BY COUNT(*) DESC
LIMIT 2
-- Minimum
SELECT MIN(score) as min_score FROM score;
-- Maximum
SELECT Max(score) as max_score FROM score;
-- Sample variance
SELECT ((SUM(score)*SUM(score) - SUM(score *score))/
((COUNT(*)-1)*(COUNT(*))))
FROM score ;
```

From table 3 we can conclude that the distribution of the score of the comments is heavily skewed. To be precise, it is right skewed and thus has positive skewness. We can further illustrate this with the help of a boxplot in figure 2.

As we can see, the median represented by the red bar lies to the extreme left of the plot. From the figure, we can also see that there are a lot of extreme values as well as some outliers.

While exploring the database, we found out that there are some authors with the name **[deleted]**. We suspect that these are the authors who have deactivated or deleted their Reddit profile and thus the database accounts them as **[deleted]**. We have 312007 comments with deleted authors which is around 7% of the database. Further we checked that around 91% of the comments with authors deleted have also the body as **[deleted]**. These cases are very difficult to analyse and thus we will ignore these cases henceforth.

```
-- no. of authors deleted
SELECT COUNT(*)
        FROM "comment" as c
        WHERE c.author == "[deleted]";
-- no. of authors and comments deleted
SELECT COUNT(*)
        FROM "comment" as c
        WHERE c.author == "[deleted]"
        AND c.'body' == "[deleted]" ;
```

We suspected that maybe the comments that were deleted and subsequently the authors who deleted their Reddit profiles, may have posted some controversial comments. But, only 1.4% of these comments were controversial. Maybe the users were bored of Reddit!

From what follows, we will dive deeper into some advanced analysis.

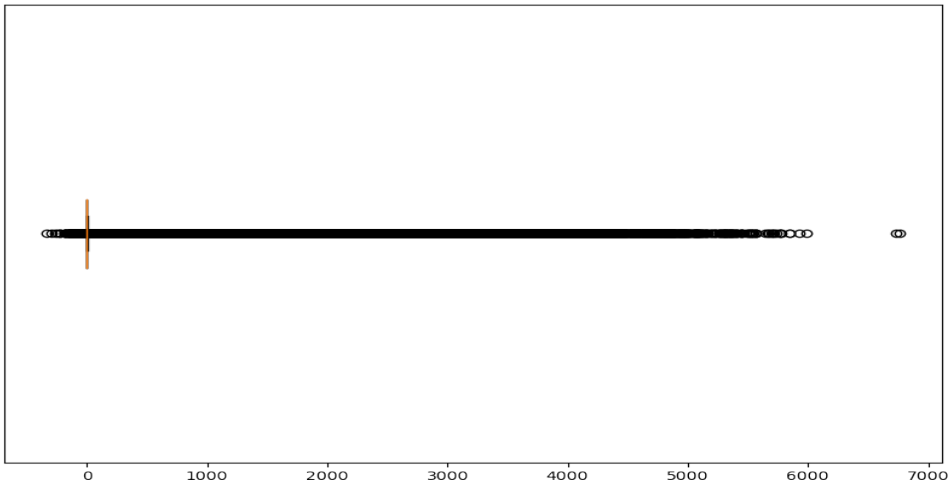

Figure 2: Boxplot for Scores

3.2 Advanced Analysis

3.2.1 Author Analysis

In the database under study, we have 57073 different authors or users of Reddit. Among these we account for all the authors who have supposedly deleted their Reddit profiles as **[deleted]**. So, we have 57072 unique authors in our database.

Our initial aim is to analyse the activeness which can be based on many criteria. First let us look at the distribution of the number of comments per author.

In our database, the minimum number of comments per author is 1 and the maximum number is 36910, after accounting for the **[deleted]** users.

```
-- Maximum number of comments per author
SELECT MAX(num)
        FROM (SELECT COUNT(com.id) as num
                        FROM "comment" as com
                        WHERE com.author <> "[deleted]"
                        GROUP BY com.author);
-- Minimum number of comments per author
SELECT MIN(num)
        FROM (SELECT COUNT(com.id) as num
                        FROM "comment" as com
                        WHERE com.author <> "[deleted]"
                        GROUP BY com.author);
```

As there is a huge gap between the minimum and the maximum number of comments per user, we were interested to look at the median of this distribution. We noted that the median was 2 which means that we have, at least **half** of the users only having 2 comments.

```
-- Median number of comments per author
SELECT num
        FROM (SELECT COUNT(com.id) as num, a.author
                                FROM "comment" as com, author as a
                                WHERE a.author = com.author
                                GROUP BY a.author)
        ORDER BY num
        LIMIT 1
```

```
        OFFSET (SELECT COUNT(*)
        FROM(SELECT COUNT(com.id) as num
                        FROM "comment" as com
                        GROUP BY com.author)) / 2;
```

As we were interested to select a group of very active users we wanted to have a deeper look into the distribution of the number of comments per user. Looking at these values for the 1000 users having the highest number of comments we noticed that the range of number of comments by a unique user is between 241 and 36910 and their cumulative distribution is 0.0017643915302198. Thus, it is very hard to choose a certain threshold, say 1000 users. According to our analysis, 1000 is not a big number to choose to specify the set of active users rather it is a big number for choosing an interesting user if we define interesting to be related to the number of comments an author/users writes.

```
-- Calculating cumulative distributions of the count of number of
-- comments for the interesting users
SELECT  num, CUME_DIST()
        OVER (ORDER BY num DESC) CumulativeDistribution
        FROM (SELECT COUNT(com.id) as num
                        FROM "comment" as com
                        WHERE com.author <> "[deleted]"
                        GROUP BY com.author)
        LIMIT 1000;
```

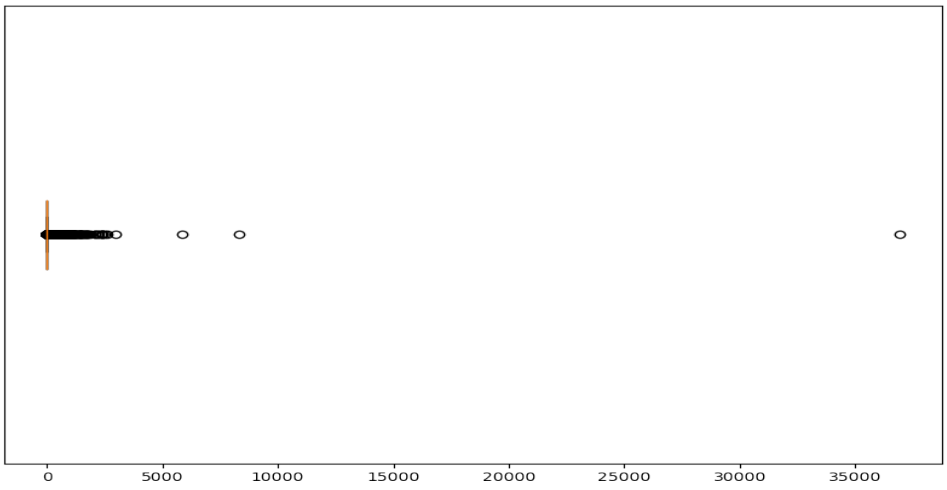

Figure 3: Boxplot for the number of comments made by the authors

As a way to understand the distribution of the comments made by the users, we plot a box plot in figure 3. We note that there are three distinct outliers in the box plot. We can tag these four outliers as the three most active users in the database, based on the number of comments an author makes.

```
SELECT COUNT(com.id) as num_comm, com.author,
                     MAX(s.ups) as max_ups,
                     MIN(s.ups) as min_ups,
                     AVG(s.ups) as avg_ups
                     FROM "comment" as com, score as s
                     WHERE s.id = com.id
                     AND com.author <> "[deleted]"
                     GROUP BY com.author
                     ORDER BY COUNT(com.id) DESC
                     LIMIT 3;
```

Author	Number of Comments	Max_score	Min_score	Avg_Score
AutoModerator*	36910	196	-90	1.00
Late_Night_Grumbler	8298	3501	-14	4.75
BiagioLargo	5843	2644	-9	3.19
-_-Equinox666-_-	2989	1476	-26	2.95

Table 4: Top 4 Authors based on number of comments

From table 4 we can see that the average score of the mentioned authors is way below the mean score of the comments in the database as mentioned in table 3. This brings us to the conclusion that defining a set of 'interesting' users is not solely based on the number of comments the author makes in the time-period considered in our study. Furthermore, we note that there is a huge gap between the number of comment made by the author *AutoModerator* and the others. This caught our attention and after investigating a bit we found out that *AutoModerator* is a system built into reddit that allows moderators to define "rules" (consisting of checks and actions) to be automatically applied to posts in their subreddit. Thus, the user having the highest number of comments is *Late_Night_Grumbler*.

We now shift our focus on the scores of the comments to check if we can find some 'interesting' users. Our aim is to see if a user has high scores for his comments and also comments frequently, thus tagging him as in *influencer*.

```
SELECT COUNT(com.id) as num_comm, com.author,
                     MAX(s.ups) as max_ups,
                     MIN(s.ups) as min_ups,
                     AVG(s.ups) as avg_ups
                     FROM "comment" as com, score as s
                     WHERE s.id = com.id
                     AND com.author <> "[deleted]"
                     GROUP BY com.author
                     ORDER BY MAX(s.ups) DESC
                     LIMIT 5;
```

Author	Number of Comments	Max_score	Min_score	Avg_Score
buckus69	118	6761	-4	66.75
47h3157	26	6736	-1	261.35
Atlasnew	29	5992	1	223.31
Blueberrytacowagon	7	5927	1	849.28
SpaizKadett	3	5849	1	1950.33

Table 5: Top 5 authors based on maximum score

From figure 2, we identified five outliers which represented the highest scores. So we wanted to identify the authors of the comments receiving these high scores. In table 5 we have some statistics for the authors with the highest maximum scores or up votes. We notice that these five authors did not make many comments in the time period under study when compared to the numbers in table 4. This brings us to the conclusion that these five users are adept at making comments that attract a lot of positive attention despite not being very active. For e.g., *SpaizKadett* posted only three comments with 5849, 1 and 1 up votes respectively. Also, the user has not received any down votes.

```
SELECT COUNT(com.id) as num_comm, com.author,
                     MAX(s.ups) as max_ups,
                     MIN(s.ups) as min_ups,
                     AVG(s.ups) as avg_ups
                     FROM "comment" as com, score as s
                     WHERE s.id = com.id
                     AND com.author <> "[deleted]"
                     GROUP BY com.author
                     ORDER BY MIN(s.ups)
                     LIMIT 5;
```

Author	Number of Comments	Max_score	Min_score	Avg_Score
harribo_11	11	37	-333	-27.45
NowSummoning	16	4	-290	-19.81
Anovan	102	325	-244	-6.18
MissyWissy	6	0	-228	-68.17
calicojackrack	2	2	-225	-111.50

Table 6: Top 5 authors based on minimum score

In a similar reasoning, we were interested in the characteristics of the authors making comments with the highest down votes. For e.g., *MissyWissy* posted 6 times in the time period under our study and never got a single up vote, poor her!

All the analysis performed in this subsection are on the user level. Next in the study, we will explore the characteristics of the comments and analyse the database at the comment level.

3.2.2 Comment Analysis

In the database under study, we have 4234970 comments. As we would like to dig deeper into the interesting ones, we decide to work with various aspects which define an 'interesting' comment.

In the database under study, every comment has an associated *parent_id*. From the course of Web Mining by Yoann Pitarch, we know that a comment can have a parent_id, and in itself the comment can be a parent_id of another comment. So, what we want to explain over here is that the relationship of the comment is similar to a family tree. We want to check if this aspect of the "comment-parent" relationship is present in this database.

$\longrightarrow$ As a first step, we wanted to list the *parent_id*s that have the highest number of comments.
$\longrightarrow$ Then, we wanted to see the characteristics of this parent_id (for e.g, the body of the comment, the number of gold reddits received, the score receiveded etc). So, we run the following query but only to get an empty output.

```
SELECT num_comments_per_parent_id, c_com.name, s.score, s.gilded
        FROM (SELECT COUNT(d.id) as num_comments_per_parent_id,
                d.parent_id as name
                        FROM depends as d
                        GROUP BY d.parent_id
                        ORDER BY COUNT(d.id) DESC
                        LIMIT 25) as c_com, score as s, "comment"
                        as com
        WHERE c_com.name = com.name
        AND s.id = com.id;
```

Further investigation, the SQL query for which is attached in the code provided, led us to conclude that the our database in designed in the structure as illustrated in figure 4. We have no information on the comments where `parent_id == name` because the parent and the comment tables are mutually exclusive.

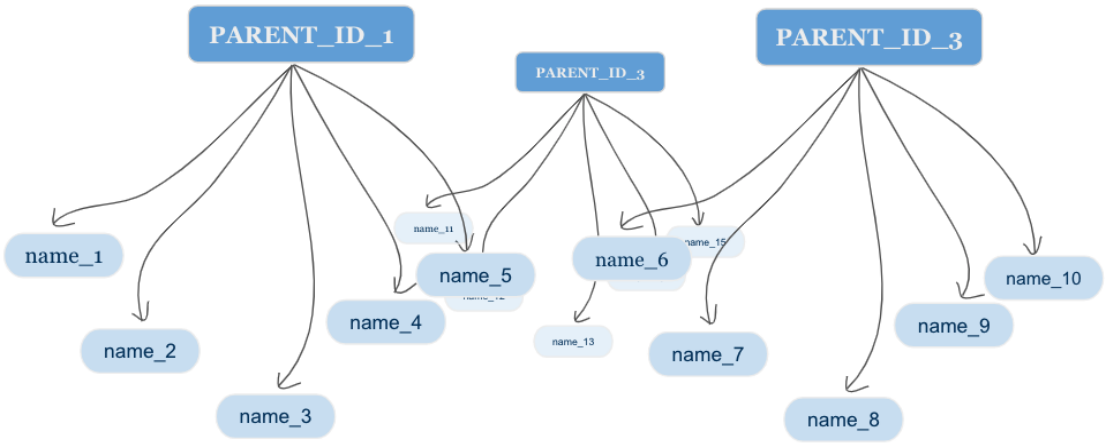

Figure 4: Link between parent_id and comments

A comment in the database under study, has various characteristics as mentioned in Section 3.1. What could be an 'interesting' comment is, if it :

→ has high score,

→ is controversial,

→ has received some reddit golds,

→ was edited,

→ has its score hidden,

→ is distinguished by moderators/admins.

As we have already explored the score dimension of the comments in the author analysis in Section 3.2.1, we decided to focus on a different aspect, the controversiality. We run an SQL query to select the controversial comments which have received gold reddits. Our motivation behind this analysis is that we want to check if some of the controversial comments attracted significant amount of attention.

```
SELECT c.id as id, c.author as author, s.score as score,
    s.gilded as nb_gilded, s.score_hidden as if_hidden,
    c.edited as if_edited, d.distinguished as if_distinguished
        FROM "comment" as c, score as s, is_distinguished as d
                WHERE c.id = s.id AND c.id = d.id
                        AND c.controversiality == 1
                        AND s.gilded > 0;
```

id_comment	Author	Score	Num_reddit_gold	if_h_score	if_edited
crnubmo	icanthinksometimes	4	1	0	1432876783
crjgyu6	trashygray	-7	1	0	1432497546
crjfzhb	PotRoastPotato	3	1	0	1432486883
crissku	columbusday	2	1	0	1432425437
criqf4n	youranimemom	2	1	0	1432415995
crbhjlt	Grokent	4	1	0	1431882443
cr9pl0f	GT2018	-8	1	0	1431671811
cr6kr5n	NilacTheGrim	10	1	0	1431488643
cr7da05	arachnopussy	1	1	0	1431486320
cr5h510	houinator	0	1	0	1431353224
cr3xezy	Oldcheese	1780	2	0	1431283638
cqxou5u	bovineblitz	0	1	0	1430766515
cqwfs8z	debatingaccount	1	1	1	1430618705
cqvu7pe	GarthVaderBlarts	4	1	0	1430580398

Table 7: Characteristics of controversial-guilded-edited comments

The output of the the SQL query gave us table 11 with 35 different comments which is attached in the Appendix. We noticed that a majority of these comments have been edited. Thus, we choose to look at the ones which were edited. From table 7, we can see that the controversial comments were mostly receiving one reddit gold and had relatively low scores. Exceptionally, the comment made by *Oldcheese* recieved 1780 upvotes and 2 reddit golds. The body of the comment is as following:

"This is actually an incredible question. I have only one kid myself, and the relationship I have with him wasn't the greatest during his youth. The day I really realized that my boy was growing up was not too long ago. I was doing some work around the yard, and he came and asked "Dad, can I ask you a question?" so I said, "Sure!" After that he asked me what I make an hour, So I told him that I make 37 euros an hour (Before taxes, ofcourse. I'm from holland) He then asked me "Can I borrow fifty bucks?" To which I replied "If the only reason you asked me about my pay is so that you can borrow some money to buy some random bottle of booze, you can get away from my house. You know I don't support that." He quietly looked at me, thinking to himself in deep thought. I just got angrier about my boy's question, How dare he ask me about money after all I did for him? but then I thought: Maybe there's something he really needs for those fifty bucks. So I say "I'm sorry son. Here's the cash." so a day later he comes back, and I ask him where he needed the cash for. his reply was "I needed to buy a good pair of pants so I can apply for this job." Needless to say, I was incredibly proud of him. From that moment I realized that my little boy had become a man. Prioritizing a job above messing around with his friends. I then asked him: Son, was there any change? He replied "Yes, About three fiddy." which he refused to give back. It was about that time I realized my son was a 7 stories tall crustacean from the paleolithic era. That damn loch ness Monster had gotten me again. Damnit, monsta, you ain't getting no three fiddy. tl;dr. Son wanted to borrow money, was for serious stuff instead of booze. Edit: Fuck you whoever gave me gold, be ashamed."

The above comment does not seem to be controversial to us. However, it has the word 'F*ck'. Our intuition is that maybe the algorithm tags comments as controversial if there are expletives present in the body of the comment. This finding inspired us to look at the words that make up the most highly scored comments. Thus, we now proceed to exploring if there is a relationship between words used in the comments and top rated comments.

We have used the top 20 high scored comments and created a word cloud. Word cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance. Significant textual data points can be highlighted using a word cloud.

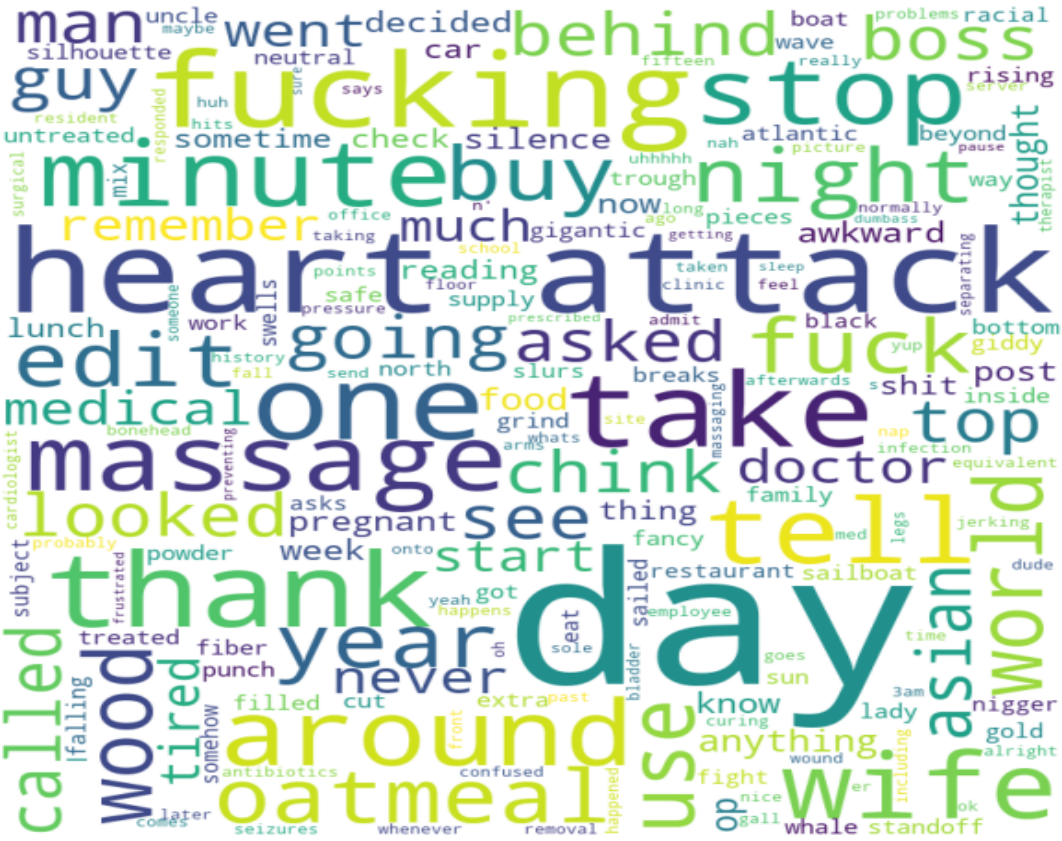

Figure 5: Word cloud for top 20 high rated comments

From figure 5, we can identify lots of words that are used in the top scored comments. Notably, we spot many words that give out a negative sentiment such as f*uck, wood (popularly used in sexual innuendos), chink, asian, racial, attack, etc. From this we can infer that the comments that address a discrimination of any kind attract a lot of attention. In table 8, we note down some statistics associated with these words.

Word	Count	Avg_Score	Max_Score	Min_Score
f*ck	175896	21.13	6736	-146
wood	13538	23.05	6736	-39
attack	12467	19.10	5845	-42
asian	5356	17.51	5927	-32
racial	1671	17.85	5927	-61
chink	108	136.5	5927	-3

Table 8: Statistics on some frequently used words

Table 8 is populated using the following kind of SQL query:

```
SELECT COUNT(com.id), AVG(s.score), MAX(s.score), MIN(s.score)
        FROM 'comment' as com, score as s
        WHERE s.id = com.id
        AND com.'body' LIKE '%wood%' ;
```

Furthermore, we want to draw your attention on a notable characteristic that we spot from figure 5. Looking at the top left of the figure, we see that the words "man" and "guy" appear more frequently than the word "lady" which is situated in the bottom-right of the figure. Although, we see the word "wife" appearing for considerable number of times. We still hold on to our belief that these comments discriminate between men and women since we know that "wife" is associated to a man and steals the real identity of being a woman.

3.2.3 Temporal Analysis

Until now, we did not look into the time dimension associated with the comments. In our database, we have the column *created_utc* which tells us the time of creation of the comment in UTC epoch-second format. We have a hypothesis that maybe the characteristics of a comment depends on the day of the week the comment is posted.

```
SELECT strftime('%w',DATETIME(c.created_utc,'unixepoch')) as Weekday,
        AVG(s.score) as score,
        COUNT(c.id) as Total_Comments,
        SUM(s.gilded),
        SUM(c.controversiality)
        FROM "comment" as c, score as s
        WHERE s.id = c.id
        GROUP BY Weekday;
```

With the help of the above SQL Query we plot some graphs. In figure 6, the bars represent the average score received by a comment and the line graph represents the total number of comments posted per day of the week.

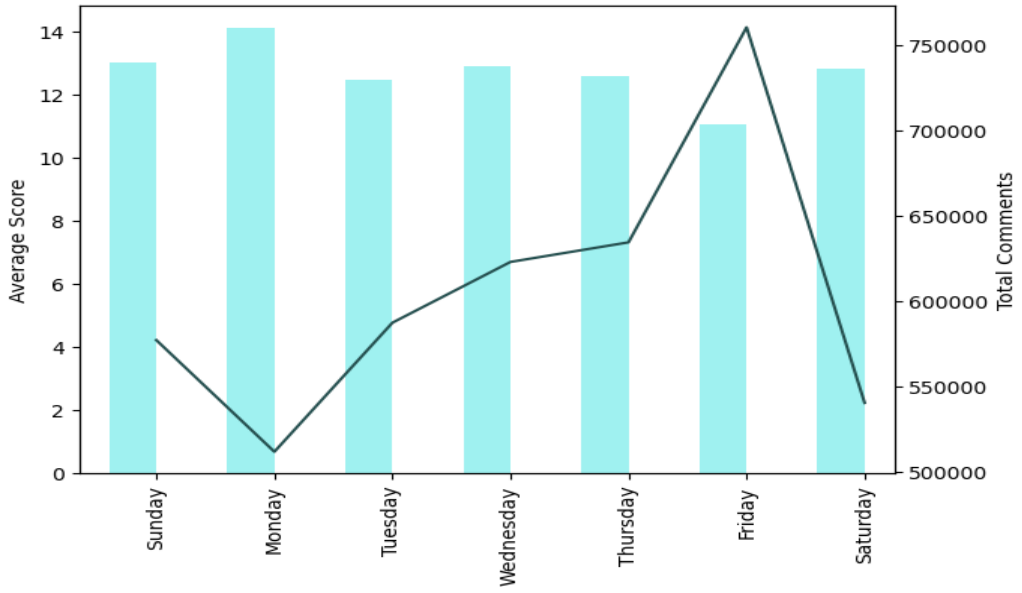

Figure 6: Average score and total comments per day of the week

We notice that Mondays have the highest average score and the lowest total comments posted. These numbers can be backed by the intuition that Monday is the first day of the week and users are generally busy, so the engagement or activity in reddit is com-

paratively lower than the other days of the week. On the other hand, the selective users that participate and post comments on Mondays are seasoned users, who make comments that attract significant attention and have high up votes, making room for the high average scores on Mondays. We also notice that Fridays have the lowest average score and the highest total comments posted. As we know, Friday is last working day of the week in most countries and users are generally in a happy-go-lucky mood! So, their participation in reddit boosts up which explains the high number of comments posted on Friday. With the increase in activity, the average score of the comment is affected and it depreciates to an all-time low in the week.

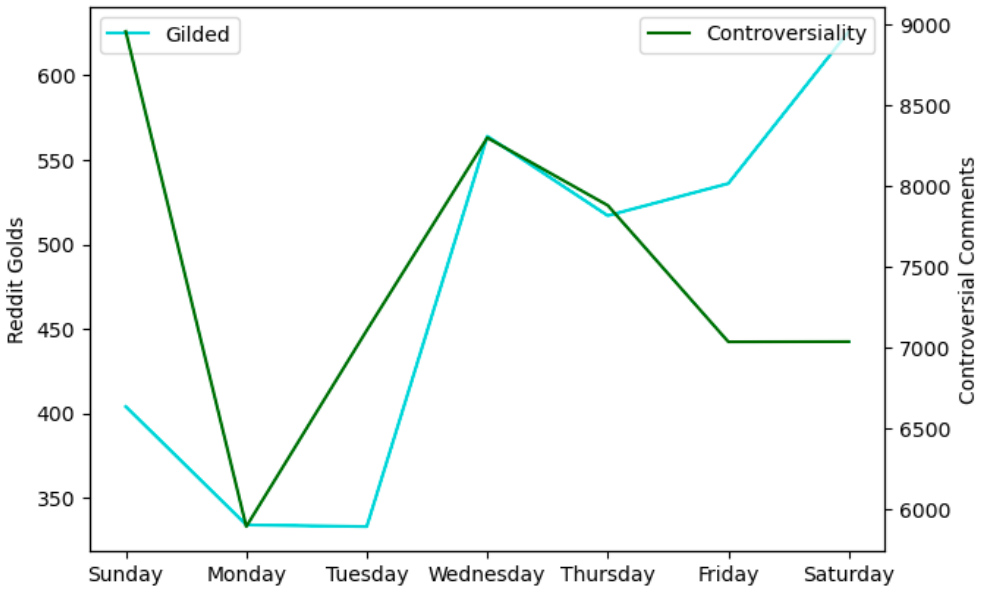

Figure 7: Reddit golds and controversial comments per day of the week

In figure 7, we have two line graphs indicating the trend of the reddit golds received by the comments and the controversial comments made over the week. We notice that on Monday, as explained earlier, the participation is low and so are the counts of reddit golds and controversial comments. The participation in reddit takes a hike from midweek and we see that the weekend has the most number of reddit golds.

4 To go further : Analysis on the comments of the Most Active users

In this section, we select and describe a sub selection of the data under study, which represents only about 5MB. Based on all the analysis conducted previously, we decided to look at the characteristics of the comments of the most active users. As we wanted to restrict the size of the database to 5MB, we took the top 7 active users. The activeness of the user is defined by the number of comments he posts. The SQL code used to select this sub part of the data is as follows:

```
.headers on
.mode csv
.output Most_Active_Users.csv

SELECT com.id, com.'body', com.author, com.controversiality,
        com.edited, s.score, s.gilded, s.score_hidden,
        d.distinguished, dep.parent_id
        FROM "comment" as com, score as s, is_distinguished as d,
            depends as dep,
                (SELECT com.author as name
                        FROM "comment" as com
                        WHERE com.author <> "AutoModerator"
                        AND com.author <> "[deleted]"
                        GROUP BY com.author
                        ORDER BY COUNT(com.id) DESC
                        LIMIT 7) as int_users
        WHERE s.id = com.id
        AND dep.id = com.id
        AND d.id = com.id
        AND com.author = int_users.name;
```

First of all, we wanted to see "what are these users talking about?". For this, we construct a word cloud from the comments these users posted. The commonly used words as seen in figure 8 are *people*, *know*, *time*, *one*, *want*, *write*, *question*, etc. Most of these words are used when we engage in a conversation in a forum to "know" something. Or maybe "gossip" about people in general. This makes absolute sense because the database under study is a extract from the **AskReddit** subreddit of reddit.

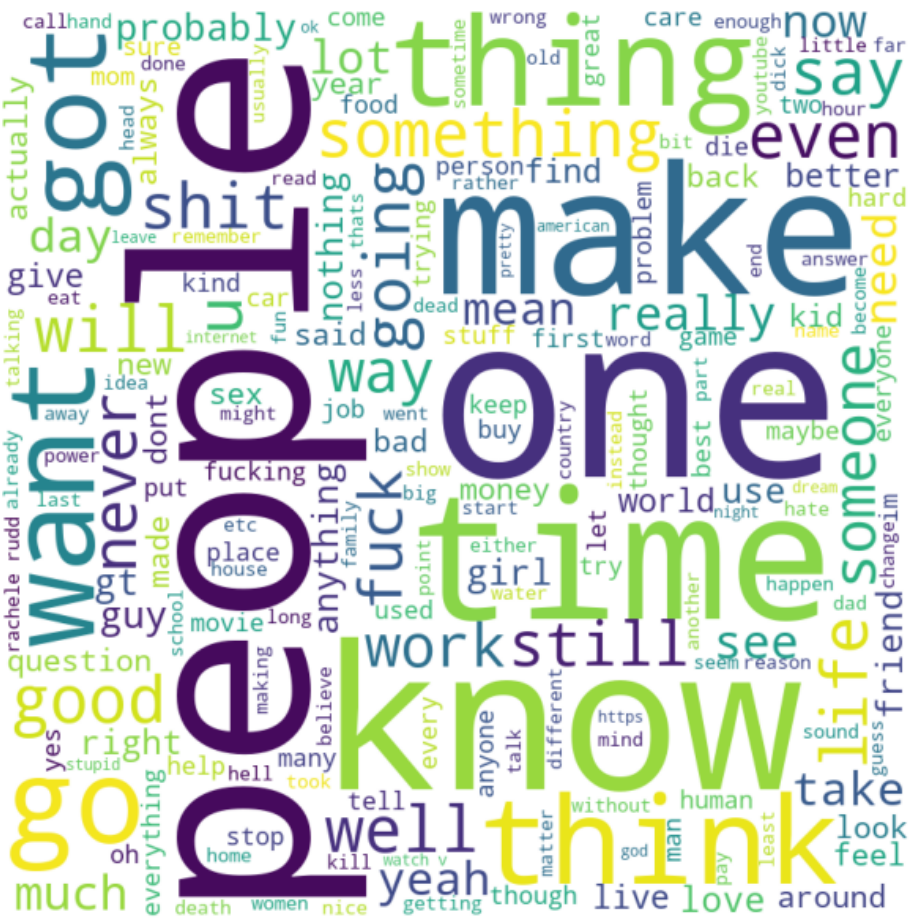

Figure 8: Word cloud of comments of the top 7 active users

Apart from the words found in their comments we also discovered an interesting pattern in the length of the comments of these users. While reading some of their comments we actually discovered that all of them have lots of comments which only consist of bigrams.

Therefore, we wanted to see whether having short comments is a shared characteristic among the most active users. To confirm our hypothesis, we ran the above query to populate table 9.

Author	Avg_length	Min_length	Max_length
Late_Night_Grumbler	11.03	1	207
BiagioLargo	17.29	1	651
-_-Equinox666-_-	7.67	1	268
KubrickIsMyCopilot	35.29	1	685
Megaross	29.47	1	403
Asdyc	11.72	1	202
NaturalInclination	16.03	1	410

Table 9: Statistics of length of comments for the 7 most active users

```
SELECT u.author , AVG(count_words.NumOfWords),
    MIN(count_words.NumOfWords), MAX(count_words.NumOfWords)
        FROM users as u,
        (SELECT u.author, CASE WHEN length(u.'body') >= 1
          THEN
    (length(u.'body') - length(replace(u.'body', ' ', '')) + 1)
          ELSE
    (length(u.'body') - length(replace(u.'body', ' ', '')))
                END as NumOfWords
                FROM users as u) as count_words
        WHERE count_words.author = u.author
        GROUP BY u.author;
```

Looking at the results we note that the average length of the comments of the most active users ranges between $[7.6, 35]$ which indicates that the comments are not very long. We also note that for each user the minimum length of a comment is 1. This confirms that there a tendency of the most active users to make shorter comments. In fact, this result is in tandem to their increased participation because writing shorter comments takes lesser time.

Finally, similar to our earlier findings we were interested in looking at the average scores and other associated characteristics of the comments of the most active users.

```
SELECT u.author, SUM(u.controversiality) as nbre_controversial,
    AVG(u.score) as average_score, MIN(u.score) as min_score,
    MAX(u.score) as max_score,
    SUM(u.gilded) as nbre_gilded,
    SUM(u.scored_hidden) as nbre_hidden
        FROM users as u
        GROUP BY u.author;
```

Looking at the results in table 10 we see some interesting traits. All these users have controversial comments. For example, *Megaross* has a total of 60 controversial comments. Besides, this user has the highest average and maximum scores when compared to the other active users. Overall, the top active users have some interesting characteristics in their comments, even if they are not the ones having the highest scored comments.

Author	Min_Score	Max_Score	Avg_Score	n_Contro	Gilded	Hid_Score
Late_Night_Grumbler	-1	96	4.75	45	1	11
BiagioLargo	-1	95	3.19	14	2	3
-_-Equinox666-_-	-1	99	2.95	49	1	1
KubrickIsMyCopilot	-1	97	2.80	57	0	7
Megaross	-1	986	21.84	60	2	0
Asdyc	-1	9	3.02	12	2	2
NaturalInclination	-1	93	3.34	13	0	4

Table 10: Statistics of length of comments for the 7 most active users

5 Conclusion

The database we built contains very interesting data that gives out a variety of interesting information on the comments found on the AskReddit subreddit or Reddit. Implementing SQL in our work was crucial in helping us answer many questions of our interest such as:

- What is inside this database ?
- Who are the interesting users ?
- What are the interesting comments ?
- On which days are people posting the most ?
- What are some comments mainly talking about ?

However, exploring the AskReddit Data is a never ending task as one will always be curious about finding out more information. A further improvement to our study would be to consider sentiment analysis of the comments. This would help us tag the comments as positive, negative or neutral. We encourage future work to explore this aspect.

6 Appendix

id_comment	Author	Score	Num_reddit_gold	if_h_score	if_edited
crnubmo	icanthinksometimes	4	1	0	1432876783
crjgyu6	trashygray	-7	1	0	1432497546
crjfzhb	PotRoastPotato	3	1	0	1432486883
crissku	columbusday	2	1	0	1432425437
criqf4n	youranimemom	2	1	0	1432415995
crbhjlt	Grokent	4	1	0	1431882443
cr9pl0f	GT2018	-8	1	0	1431671811
cr6kr5n	NilacTheGrim	10	1	0	1431488643
cr7da05	arachnopussy	1	1	0	1431486320
cr5h510	houinator	0	1	0	1431353224
cr3xezy	Oldcheese	1780	2	0	1431283638
cqxou5u	bovineblitz	0	1	0	1430766515
cqwfs8z	debatingaccount	1	1	1	1430618705
cqvu7pe	GarthVaderBlarts	4	1	0	1430580398
cqvxzni	TotesHuman	190	1	0	0
cr0bs8e	Lunux	21	1	0	0
cr2ip1b	_iPood_	3	1	0	0
cr2ujc9	HornyBillyGoat	496	1	0	0
cr3i099	DmitryTheSheep	-1	1	0	0
cr3i2ag	kyle8998	0	1	0	0
cr4qt3w	coolcrate	2	1	0	0
cr57n9k	stuck_at_starbucks	3	1	0	0
crby1bl	borick	3	1	0	0
crc3fo0	superwinner	2	1	0	0
cre7cdi	blades_of_furry	-4	1	0	0
crgc1vl	iqtestforhiring	1	1	0	0
crhc36u	iam4real	-10	1	0	0
criqf42	buffdude1100	2	1	0	0
criqlvq	jkmartindale	-3	1	0	0
crj3lz2	Algamain	-49	1	0	0
crjw06t	JustMe80	1	1	0	0
crnuw7c	Reddit-And-Wept	1	1	0	0
crpn4br	I_Like_Quiet	-6	1	0	0
crq0hi2	PhoenixDays	214	1	0	0

Table 11: Characteristics of controversial-guilded-edited comments