Contents

Unnormalized Form of database

bad_shorter_table (

created_at

tweet_id PRIMARY KEY

in_reply_to_screen_name

in_reply_to_status_

in_reply_to_user_

retweet_count

tweet_source

retweet_of_

hashtag1

hashtag2

hashtag3

hashtag4

hashtag5

hashtag6

user_id

user_name

user_screen_name

user_location

user_utc_offset

user_time_zone

user_followers_count

user_friends_count

user_lang character

user_description

user_status_count

user_created_at

)

We are making the assumptions that each tweet has a unique tweet_id, each user has unique user_id. Hence, we can use them as a primary key.

First Normal Form

A database table is considered to be in first normal form when it meets the following criteria:

Each table cell should contain only atomic values (i.e., indivisible and cannot be further broken down into smaller values).

Each column in a table should have a unique name.

Each column in a table should contain data of the same data type.

Each row in a table should be uniquely identifiable.

We have observed above database; since there are no multi valued columns hence we can say that it is already in first Normal Form.

1NF Explanation

To convert the table into 1NF, we need to remove repeating groups of columns. In this case, the columns hashtag1, hashtag2, hashtag3, hashtag4, hashtag5, and hashtag6 represent a repeating group, which should be split into a separate table.

We can create a new table named "tweet_hashtag" with the columns "tweet_id" and "hashtag", where "tweet_id" is a foreign key referencing the "tweet_id" column in the original table, and "hashtag" is a varchar(144) column representing a single hashtag.

Functional Dependencies (FD):

Following set of functional dependencies are possible after splitting the table:

1. tweet_id -> created_at, text, in_reply_to_screen_name, in_reply_to_status_id, in_reply_to_user_id, retweet_count, tweet_source, retweet_of_tweet_id, user_id, user_name, user_screen_name, user_location, user_utc_offset, user_time_zone, user_followers_count, user_friends_count, user_lang, user_description, user_status_count, user_created_at

2. tweet_id -> hashtag1, hashtag2, hashtag3, hashtag4, hashtag5, hashtag6

Second Normal Form

Based on the table definition and the functional dependencies identified for 1NF, we can see that the "bad_shorter_table" is not fully normalized because the user-related attributes depend on the tweet_id, which is not a candidate key for them. To normalize the table into 2NF, we can split the table into two tables, one for tweet-related attributes and another for user-related attributes.

The following functional dependency breaks the rule of second normal form:

tweet_id -> user_name, user_screen_name, user_location, user_utc_offset, user_time_zone, user_followers_count, user_friends_count, user_lang, user_description, user_status_count, user_created_at

Third Normal Form

To convert the given table to 3NF, we need to eliminate any transitive dependencies and group the columns that belong together into separate tables. In the bad table, we can observe that user attributes such as "user_name ", "user_screen_name" functionally depends upon "user_id", and "user_id " functionally depends upon "tweet_id ". This functional dependency breaks the third normal form rule.

tweet_id -> user_id

user_id -> user_name, user_screen_name

After splitting the bad table for user attributes and tweet attributes, this transitivity can be eliminated.

Following set of functional dependencies are possible after splitting the table:

user_id -> user_name, user_screen_name, user_location, user_utc_offset, user_time_zone, user_followers_count, user_friends_count, user_lang, user_description, user_status_count, user_created_at

tweet_id -> created_at, text, in_reply_to_screen_name, in_reply_to_status_id, in_reply_to_user_id, retweet_count, tweet_source, retweet_of_tweet_id, user_id

tweet_id -> hashtag1, hashtag2, hashtag3, hashtag4, hashtag5, hashtag6

New Table Design

users (
    user_ID PRIMARY KEY,
    user_name
    user_screen_name
    user_location
    user_utc_offset
    user_time_zone
    user_followers_count
    user_friends_count
    user_lang character
    user_description
    user_status_count
    user_created_at
)

tweets (
    Tweet_ID PRIMARY KEY,
    User_ID FOREIGN KEY REFERENCES users(user_ID)
    in_reply_to_user_id
    created_at
    Text
    in_reply_to_screen_name
    in_reply_to_status_id
    retweet_count
    tweet_source
    retweet_of_tweet_id
)

hashtags (
    Hashtag_ID PRIMARY KEY
    Hashtag
)

```
tweet_hashtag (
    tweet_hashtag_id PRIMARY KEY
    tweet_id FOREIGN KEY REFERENCES tweet(tweet_id)
    hashtag_id FOREIGN KEY REFERENCES hashtags(hashtag_id)
)
```

Query Outputs

1.1

```
postgres=# -- Section 1
postgres=# -- How many tweets are there in total?
postgres=# SELECT COUNT(*) AS total_tweets FROM Tweets;
 total_tweets
--------------
       110573
(1 row)


postgres=# -- Every row represents a unique tweet, so number of rows is number of tweets
```

1.2

```
postgres=# -- How are these tweets distributed across languages? Write a query that shows, for every
postgres=# -- language( user_lang ) the number of tweets in that language.
postgres=# SELECT u.user_lang AS language, COUNT(*) AS num_tweets FROM Tweets t JOIN Users u ON t.User_ID = u.User_ID GROUP BY u.user_lang;
 language | num_tweets
----------+------------
 ar       |       1405
 ca       |          7
 cs       |          2
 de       |         75
 el       |          2
 en       |      74076
 es       |      17910
 eu       |          1
 fi       |          1
 fil      |          9
 fr       |        231
 hu       |          1
 id       |       1423
 it       |         26
 ja       |       9861
 ko       |        691
 msa      |         14
 nl       |         61
 no       |          1
 pl       |          4
 pt       |       3977
 ru       |        396
 sv       |          2
 th       |        233
 tr       |        123
 ur       |          1
 zh-cn    |         26
 zh-tw    |         14
(28 rows)


postgres=# -- Performing join on tweets and users table using User_id, grouping by user language, the number of rows
postgres=# -- will be the number of tweets for that language
```

1.3

```
postgres=# -- Compute, for each language, the fraction of total tweets that have that language setting, as well as
postgres=# -- the fraction of the number of users that have that language setting.
postgres=# SELECT u.user_lang, COUNT(*) AS num_tweets, COUNT(DISTINCT t.User_ID) AS num_users, COUNT(*)::float / (SELECT COUNT(*) FROM Tweets)::float AS twe
et_fraction, COUNT(DISTINCT t.User_ID)::float / (SELECT COUNT(DISTINCT User_ID) FROM Users)::float AS user_fraction FROM Tweets t JOIN Users u ON t.User_ID
= u.User_ID GROUP BY u.user_lang;
 user_lang | num_tweets | num_users |     tweet_fraction     |     user_fraction
-----------+------------+-----------+------------------------+------------------------
 ar        |       1405 |      1328 |   0.012706537762383221 |   0.012795560094810475
 ca        |          7 |         7 |  6.330659383393776e-05 |  6.7446476403368470e-05
 cs        |          2 |         2 | 1.8087598238267933e-05 | 1.9270421829533848e-05
 de        |         75 |        71 |  0.0006782849339350475 |  0.0006840999749484516
 el        |          2 |         2 | 1.8087598238267933e-05 | 1.9270421829533848e-05
 en        |      74076 |     69278 |     0.6699284635489676 |     0.6675081417532229
 es        |      17910 |     16711 |    0.16197444222368934 |    0.16101400959667006
 eu        |          1 |         1 |  9.043799119133966e-06 |  9.6352109147669246e-06
 fi        |          1 |         1 |  9.043799119133966e-06 |  9.6352109147669246e-06
 fil       |          9 |         9 |  8.139419207220569e-05 |  8.6716898232902320e-05
 fr        |        231 |       221 |   0.002089117596519946 |   0.002129381612163490
 hu        |          1 |         1 |  9.043799119133966e-06 |  9.6352109147669246e-06
 id        |       1423 |      1360 |   0.012869326146527634 |   0.013103886844083017
 it        |         26 |        25 | 0.0002351387770974831 2 |  0.0002408802728691731
 ja        |       9861 |      9596 |    0.08918090311378003 |    0.092459483938103410
 ko        |        691 |       653 |    0.006249265191321570 |  0.0062917927273428015
 msa       |         14 |        13 | 0.0001266131876678755 2 | 0.0001252577418919700 1
 nl        |         61 |        60 |  0.000551617746267171 9 |  0.0005781126548860154
 no        |          1 |         1 |  9.043799119133966e-06 |  9.6352109147669246e-06
 pl        |          4 |         4 | 3.6175196476535865e-05 | 3.8540843659067696e-05
 pt        |       3977 |      3684 |    0.035967189096795780 |    0.035496117010001350
 ru        |        396 |       382 |  0.003581344451177050 3 |  0.0036806505694409650
 sv        |          2 |         2 | 1.8087598238267933e-05 | 1.9270421829533848e-05
 th        |        233 |       217 |   0.002107205194758214 |  0.0020908407685044225
 tr        |        123 |       121 |   0.001112387291653478 |  0.0011658605206867978
 ur        |          1 |         1 |  9.043799119133966e-06 |  9.6352109147669246e-06
 zh-cn     |         26 |        21 | 0.0002351387770974831 2 | 0.0002023394292101054 2
 zh-tw     |         14 |        14 | 0.0001266131876678755 2 | 0.0001348929528067369 5
(28 rows)


postgres=# -- Joining tweets and users table with User_id, grouped by user_lang. Calculating fraction of all tweet
postgres=# -- belonging to each language and fraction of all users posted tweet in each language
```

2.1, 2.2, 2.3

```
postgres=# -- Section 2
postgres=# -- What fraction of the tweets are retweets?
postgres=# SELECT COUNT(*)::float / (SELECT COUNT(*) FROM Tweets)::float AS retweet_fraction FROM Tweets WHERE retweet_count > 0;
  retweet_fraction
--------------------
 0.3307859965814439
(1 row)


postgres=# -- Dividing tweets that are retweeted with total number of tweets.
postgres=#
postgres=# -- Compute the average number of retweets per tweet.
postgres=# SELECT AVG(retweet_count) AS avg_retweets FROM Tweets;
     avg_retweets
----------------------
 121.1948396082226222
(1 row)


postgres=# -- Averaging retweet_count column to find average retweet count
postgres=#
postgres=# -- What fraction of the tweets are never retweeted?
postgres=# SELECT COUNT(*)::float / (SELECT COUNT(*) FROM Tweets)::float AS no_retweet_fraction FROM Tweets WHERE retweet_count = 0;
 no_retweet_fraction
---------------------
  0.6692140034185561
(1 row)


postgres=# -- Dividing tweets that are never tweeted with total number of tweets
```

2.4

```
postgres=# -- What fraction of the tweets are retweeted fewer times than the average number of retweets (and what
postgres=# -- does this say about the distribution)?
postgres=# WITH avg_retweets AS ( SELECT AVG(retweet_count) AS avg_retweets FROM Tweets )
postgres=# SELECT COUNT(*)::float / (SELECT COUNT(*) FROM Tweets)::float AS retweeted_less_than_avg_fraction FROM Tweets WHERE retweet_count < (SELECT avg_r
etweets FROM avg_retweets);
 retweeted_less_than_avg_fraction
----------------------------------
               0.936205041013629
(1 row)


postgres=# -- Calculated the average number of retweets, then counted number of tweets that are less retweeted than
postgres=# -- average, then divided the count with total number of tweets
postgres=# -- The fraction is 0.94, that means the distribution is right skewed
```

3.1, 3.2

```
postgres=# -- Section 3
postgres=# -- What is the number of distinct hashtags found in these tweets?
postgres=# SELECT COUNT(DISTINCT Hashtag) AS num_distinct_hashtags FROM Hashtags;
 num_distinct_hashtags
-----------------------
                 10158
(1 row)


postgres=# -- Distinct word will not count the repeated hashtag
postgres=#
postgres=# -- What are the top ten most popular hashtags, by number of usages?
postgres=# SELECT Hashtags.Hashtag, COUNT(tweet_hashtag.tweet_id) AS usage_count FROM Hashtags INNER JOIN tweet_hashtag ON Hashtags.Hashtag_ID = tweet_hasht
ag.hashtag_id GROUP BY Hashtags.Hashtag ORDER BY usage_count DESC LIMIT 10;
         hashtag          | usage_count
--------------------------+-------------
 ReasonsIFailAtBeingAGirl |         467
 RED                      |         240
 oomf                     |         190
 HonestyHour              |         172
 TeamFollowBack           |         139
 EresGuapaSi              |         130
 10PeopleYouTrulyLove     |         126
 TweetLikeAGirl           |          98
 ImSingleBecause          |          97
 WeAllGotThatOneFriend    |          96
(10 rows)


postgres=# -- Joined the table hashtags and tweet_hashtag on Hashtag_id, grouped by hashtag and ordered by the
postgres=# -- usage count, printed in descending order
postgres=#
```

3.3

```
postgres=# -- Write a query giving, for each language, the top three most popular hashtags in that language.
postgres=# WITH hashtag_counts AS ( SELECT Users.user_lang, Hashtags.Hashtag, COUNT(tweet_hashtag.tweet_id) AS usage_count, ROW_NUMBER() OVER (PARTITION BY Users.user_lang ORDER BY COUNT(tweet_hashtag.tweet_id) DESC) AS rank FROM Users JOIN Tweets ON Users.User_ID = T
weets.User_ID JOIN tweet_hashtag ON Tweets.Tweet_ID = tweet_hashtag.tweet_id JOIN Hashtags ON tweet_hashtag.hashtag_id = Hashtags.Hashtag_ID GROUP BY Users.user_lang, Hashtags.Hashtag)
postgres-# SELECT user_lang, Hashtag, usage_count FROM hashtag_counts WHERE rank <= 3 ORDER BY usage_count DESC;
 user_lang |              hashtag              | usage_count
-----------+-----------------------------------+-------------
 en        | ReasonsIFailAtBeingAGirl          |         459
 en        | RED                               |         214
 en        | oomf                              |         190
 es        | EresGuapaSi                       |         120
 es        | LoAdmito                          |          50
 es        | SoloElVenezolano                  |          42
 ja        | countkun                          |          38
 ja        | sougofollow                       |          27
 ja        | follownejp                        |          20
 pt        | YGDONTFORGETBRAZIL                |          18
 ar        | ╪┘ê╪┐ê╪*                              |             18
 ar        | ╪╫╪┐╪»_╪¿╪┘ê╪┐╪                                |          13
 id        | TeamFollowBack                    |          11
 ar        | ╪°┘ä┘ä┘ê┘ê╪                              |           11
 ru        | gameinsight                       |          11
 ru        | androidgames                      |          10
 id        | JJ                                |           9
 ru        | Android                           |           8
 pt        | RT                                |           7
 id        | TFBJP                             |           7
 pt        | 7PessoasQueSintoCiumes            |           6
 tr        | C├╝bbeliAhmetHocaya├ûzg├╝rl├╝k─▒stiyoruz |            4
 ko        | δ òΩ┐╛ϕÿ£                           |          4
 th        | α╣üα╕«α╕Üα╕┤α╕ùα╕Üα╕Ñα╕▒α╕ öα╣âα╕ùα╕ ¢α╣ì                         |            4
 tr        | MetrisB├╝yleBiriniG├╢rmedi         |          4
 tr        | METR─ S─ NG├£L├£                     |            4
 ko        | δ°ÿ∞ùÉδôáδ¥┤                          |            3
 fr        | RT                                |           3
 fr        | tvanouvelles                      |           3
 ko        | mrpizza                           |           3
 th        | imsple                            |           2
 de        | Hamburg                           |           2
 fr        | MashupImprobable                  |           2
 th        | VRZO                              |           2
 nl        | TEAMFOLLOWWACK                    |           1
 nl        | Truste                            |           1
 de        | Thesis                            |           1
 de        | tracking                          |           1
 cs        | fermentation                      |           1
 cs        | continuous                        |           1
 ca        | SoyelIngrato                      |           1
 zh-cn     | iPad0óæ0ûô                          |          1
 ca        | EseMomentoEnElQue                 |           1
 zh-cn     | iphone                            |           1
 zh-tw     | sm18972686                        |           1
 it        | follow                            |           1
 fil       | food                              |           1
 it        | Adieu                             |           1
 it        | sapevatelo                        |           1
 fil       | 10TwitterCrushes                  |           1
 fil       | noexcuse                          |           1
 el        | soundcloud                        |           1
 zh-cn     | Lion                              |           1
 nl        | actrices                          |           1
(54 rows)
```

4.1,4.2

```
postgres=# -- Section 4
postgres=# -- How many tweets are neither replies, nor replied to?
postgres=# SELECT COUNT(*) FROM Tweets WHERE in_reply_to_status_id IS NULL AND in_reply_to_user_id IS NULL;
 count
-------
 84798
(1 row)


postgres=# -- Counting the tweets that are not a reply to another tweet, and also has not received any reply
postgres=#
postgres=# -- If a user user1 replies to another user user2, what is the probability that they have the same
postgres=# -- language setting?
postgres=# SELECT COUNT(*) FILTER (WHERE Users1.user_lang = Users2.user_lang) AS same_lang_count, COUNT(*) AS total_count, COUNT(*) FILTER (WHERE Users1.use
r_lang = Users2.user_lang)::numeric / COUNT(*)::numeric AS probability FROM Tweets AS Replies INNER JOIN Users AS Users1 ON Replies.User_ID = Users1.User_ID
 INNER JOIN Users AS Users2 ON Replies.in_reply_to_user_id = Users2.User_ID WHERE Replies.in_reply_to_user_id IS NOT NULL AND Replies.in_reply_to_status_id
IS NOT NULL;
 same_lang_count | total_count |      probability
-----------------+-------------+------------------------
            3191 |        3451 | 0.92465951898000579542
(1 row)


postgres=# -- Counting the number of replies where the language of the user who replied matches the language
postgres=# -- of the user being replied to, and dividing that count by the total number of replies
```