SQL Problem

The code for the SQL part is shared in SolutionCode.sql

Assuming SQLite

1) Please aggregate the number of connections that each user has

The users having ID 2 and 4 have the maximum number of connections.

2) How do we see how many mutual connections each pair of users have?

The pair 6-1 has 2 mutual connections

3) Any other comments on this problem - are there better ways to store the data?

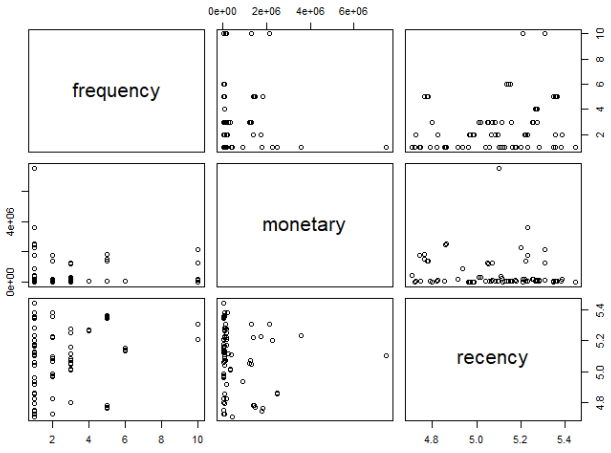
I think with the usage or Oracle DB, making use of concat and like (%) function, the data can be stored and accessed easily. In that manner every user will have one row assigned to it and two columns, one column for it's ID and another column detailing the connection ID separated by comma (using concat) as follows:

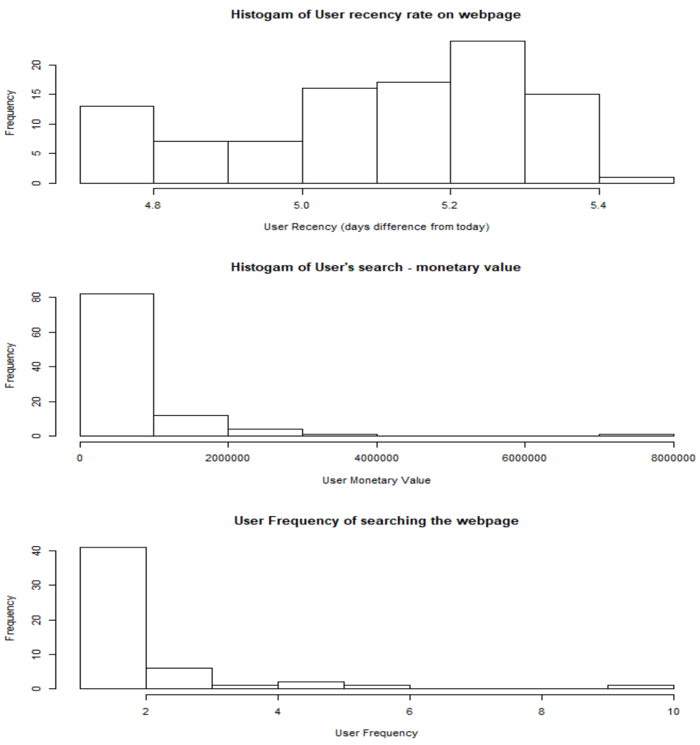
User	Connections
1	2, 3, 7, 10
2	1, 5, 8
And so on …	

Another way is the way depicted in the solution for second question, i.e. by creation of a friendship table and adding a friendship ID in it. In that case if A is friends with B, we will have two columns, one having friendship ID and another having the connection "A-B". It is important to keep some sign (in this case dash sign) in between so as to be able to not confuse the User IDs.

Modelling Problem

The solution is saved in SolutionCode.R
RFM (recency, frequency, monetary) analysis has been done.
The resulting graphs have been stored in RFM.png and RFM_histogram.png.

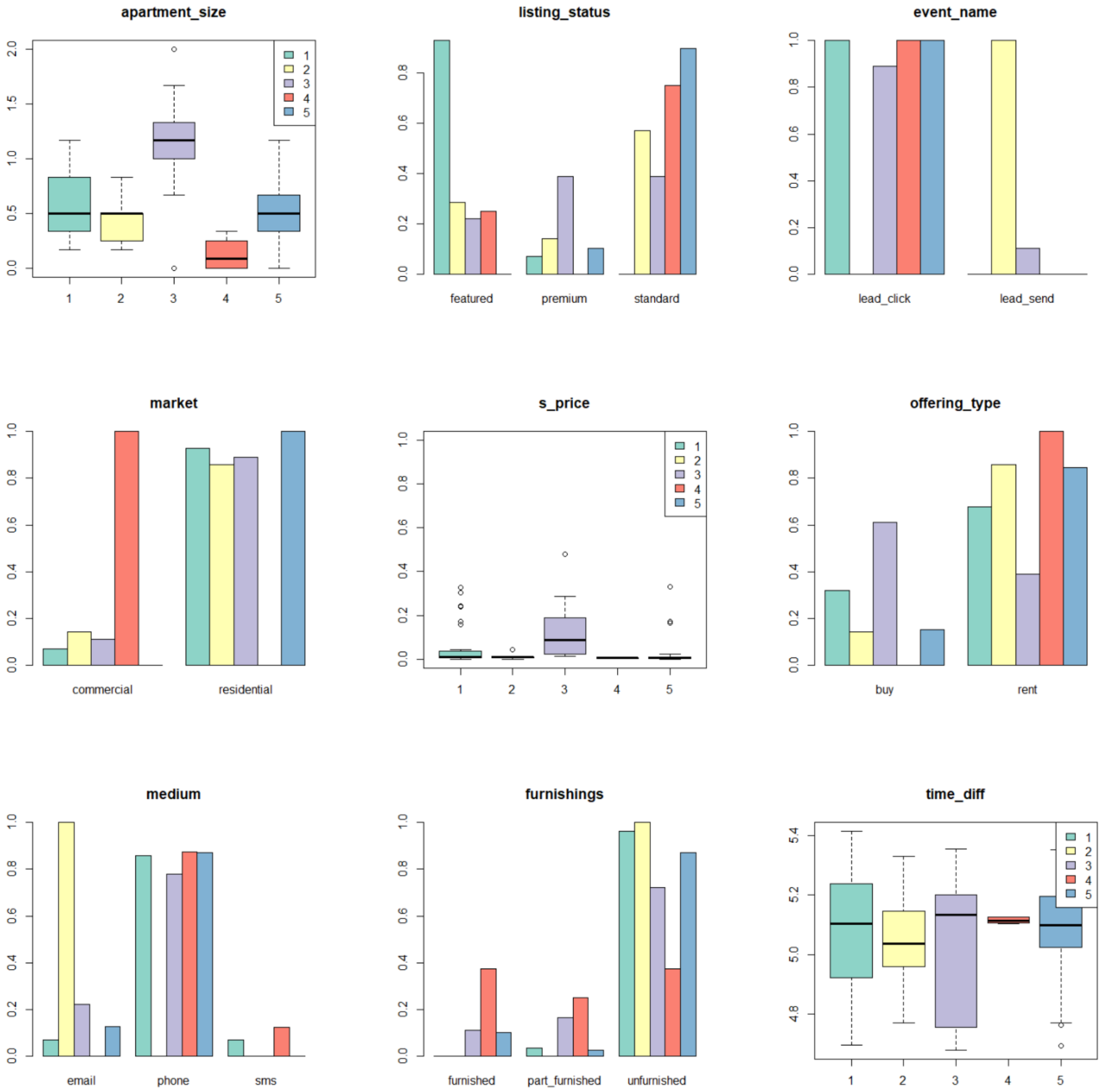




Partitioning clustering has been done on leads, and 5 types of groups have been obtained as a result, featuring:

Group	Event	medium	offering_type	property_type_id	furnishings	market	listing_status
1	lead_click	phone	rent	1	unfurnished	residential	featured
2	lead_send	email	rent	1	unfurnished	residential	standard
3	lead_click	phone	buy	35	unfurnished	residential	don't care
4	lead_click	phone	rent	4	don't care	commercial	standard
5	lead_click	phone	rent	1	unfurnished	residential	standard

Other factors like price and size of the apartment as also been considered. The graph for the same is stored in clusters_5.png



2. Now that you have created your segmentation algorithm, please describe how you might go about putting this into production so that it's regularly updated

Based on OS and availability of the databases, the RFM script can be scheduled using Task Scheduler & shiny

3. Aside from the features you create from the provided data, what else do you think could support an effective segmentation project from a tracking/features point of view?

From tracking more features pov, looking into features such as "apartment search or room search" or "inbox when similar listings come" can also be added.

Metrics Problem

1. We recently changed our search function, and implemented a new ranking algorithm for our properties. Please suggest some metrics that may be useful for assessing whether a change to our new ranking algorithm was successful. Why are these useful metrics?

Trying to decipher the problem statement

- Assessing ranking algorithm for properties => properties are given ranks from best to worst based on a search algorithm
- My doubt is who is the "end user" of the ranking algorithm, based on that metrics can be designed to access the effectiveness of the algorithm

Assuming ranking algorithm rates properties based on their ease of being searched by the user vising the webpage. In that case, the number of clicks on a page after the search will be one metric to tell the effectiveness of the algorithm.

In another assumption, if ranking algorithm rates properties based on their ease of being sold, then looking at the type of searches and the type of properties, a demand-supply graph can be tried.

2. One metric we check is the average time to lead after a first page view. Looking back from today (14th Nov) we get a chart that looks like this:

The graph is not clear. Below are some of the conclusions I tried making from the data:

- Average number of days to book on Oct 14th were more than the average number of days to book Nov 11th.
- For some given number of leads, average days to book decreased, as the leads were converted with time.