```
#please import the SQL file “text mining.sql” to your MySQL working space
#step 1: remove punctuation and remove extra space
update textmining_data set title = REPLACE(title,"`","");
update textmining_data set title = REPLACE(title,"~","");
update textmining_data set title = REPLACE(title,"!","");
update textmining_data set title = REPLACE(title,"@","");
update textmining_data set title = REPLACE(title,"#","");
update textmining_data set title = REPLACE(title,"$","");
update textmining_data set title = REPLACE(title,"%","");
update textmining_data set title = REPLACE(title,"^","");
update textmining_data set title = REPLACE(title,"&","");
update textmining_data set title = REPLACE(title,"*","");
update textmining_data set title = REPLACE(title,"(","");
update textmining_data set title = REPLACE(title,")","");
update textmining_data set title = REPLACE(title,"-","");
update textmining_data set title = REPLACE(title,"_","");
update textmining_data set title = REPLACE(title,"=","");
update textmining_data set title = REPLACE(title,"+","");
update textmining_data set title = REPLACE(title,"{","");
update textmining_data set title = REPLACE(title,"}","");
update textmining_data set title = REPLACE(title,"[","");
update textmining_data set title = REPLACE(title,"]","");
update textmining_data set title = REPLACE(title,"|","");
update textmining_data set title = REPLACE(title,";","");
update textmining_data set title = REPLACE(title,":","");
update textmining_data set title = REPLACE(title,"'","");
update textmining_data set title = REPLACE(title,"<","");
update textmining_data set title = REPLACE(title,",","");
update textmining_data set title = REPLACE(title,">","");
update textmining_data set title = REPLACE(title,".","");
update textmining_data set title = REPLACE(title,"/","");
update textmining_data set title = REPLACE(title,"?","");
update textmining_data set title = REPLACE(title,"\\","");
update textmining_data set title = REPLACE(title,"\"","");
update textmining_data set title = REPLACE(title,"!"","");
update textmining_data set title = REPLACE(title,"'","");
update textmining_data set title = REPLACE(title,",","");
update textmining_data set title = REPLACE(title,"?","");
update textmining_data set title = REPLACE(title,"&","");

update textmining_data set title = REPLACE(title,".","");
update textmining_data set title = REPLACE(title,"[","");
update textmining_data set title = REPLACE(title,"]","");
update textmining_data set title = REPLACE(title,"-","");
update textmining_data set title = REPLACE(title,"*","");
update textmining_data set title = REPLACE(title,"/","");

# detect the number of the words.
SELECT id, title, LENGTH(title) - LENGTH(REPLACE(title, ' ', '')) +1 AS
word_count FROM textmining_data ORDER BY word_count DESC;
ALTER TABLE textmining_data ADD COLUMN `no_of_words` int;
UPDATE textmining_data SET no_of_words = LENGTH(title) -
LENGTH(REPLACE(title, ' ', '')) +1;
# you will find the maximal amount of the words is 10, as next step, we
retrieve the individual words to be new columns.
```

```
# we have explained the meaning of the following function in the previous
lecture when introducing SUBSTRING_INDEX function.
SELECT id, title, no_of_words, SUBSTRING_INDEX(title, ' ', 1) AS word1,
SUBSTRING_INDEX(SUBSTRING_INDEX(title, ' ', 2), ' ', -1)  AS word2,
SUBSTRING_INDEX(SUBSTRING_INDEX(title, ' ', 3), ' ', -1)  AS word3,
SUBSTRING_INDEX(SUBSTRING_INDEX(title, ' ', 4), ' ', -1)  AS word4,
SUBSTRING_INDEX(SUBSTRING_INDEX(title, ' ', 5), ' ', -1)  AS word5,
SUBSTRING_INDEX(SUBSTRING_INDEX(title, ' ', 6), ' ', -1)  AS word6,
SUBSTRING_INDEX(SUBSTRING_INDEX(title, ' ', 7), ' ', -1)  AS word7,
SUBSTRING_INDEX(SUBSTRING_INDEX(title, ' ', 8), ' ', -1)  AS word8,
SUBSTRING_INDEX(SUBSTRING_INDEX(title, ' ', 9), ' ', -1)  AS word9,
SUBSTRING_INDEX(SUBSTRING_INDEX(title, ' ', 10), ' ', -1)  AS word10
FROM textmining_data
# Nonetheless, we find that there is a problem that SUBSTRING_INDEX returns
all the values if delimiter cannot be found.

# Can you figure out a solution for this?
# submit your answer at Presmo.
```

```
create table word_table AS (SELECT id, title, no_of_words,
SUBSTRING_INDEX(title, ' ', 1) AS word1,
case when no_of_words > 1 then SUBSTRING_INDEX(SUBSTRING_INDEX(title, ' ',
2), ' ', -1) end AS word2,
case when no_of_words > 2 then SUBSTRING_INDEX(SUBSTRING_INDEX(title, ' ',
3), ' ', -1) end AS word3,
case when no_of_words > 3 then SUBSTRING_INDEX(SUBSTRING_INDEX(title, ' ',
4), ' ', -1) end AS word4,
case when no_of_words > 4 then SUBSTRING_INDEX(SUBSTRING_INDEX(title, ' ',
5), ' ', -1) end AS word5,
case when no_of_words > 5 then SUBSTRING_INDEX(SUBSTRING_INDEX(title, ' ',
6), ' ', -1) end AS word6,
case when no_of_words > 6 then SUBSTRING_INDEX(SUBSTRING_INDEX(title, ' ',
7), ' ', -1) end AS word7,
case when no_of_words > 7 then SUBSTRING_INDEX(SUBSTRING_INDEX(title, ' ',
8), ' ', -1) end AS word8,
case when no_of_words > 8 then SUBSTRING_INDEX(SUBSTRING_INDEX(title, ' ',
9), ' ', -1) end AS word9,
case when no_of_words > 9 then SUBSTRING_INDEX(SUBSTRING_INDEX(title, ' ',
10), ' ', -1) end AS word10
FROM textmining_data);

# Nonetheless, the obtained table is very different from what we have seen in
the example of the association analysis. What to do next?
# submit your answer at presmo.

# Reflection would the output be difference using union, not union all.
create table word_list
(SELECT id, word1 FROM word_table
UNION ALL
SELECT id, word2 FROM word_table
UNION ALL
SELECT id, word3 FROM word_table
```

```
UNION ALL
SELECT id, word4 FROM word_table
UNION ALL
SELECT id, word5 FROM word_table
UNION ALL
SELECT id, word6 FROM word_table
UNION ALL
SELECT id, word7 FROM word_table
UNION all
SELECT id, word8 FROM word_table
UNION ALL
SELECT id, word9 FROM word_table
UNION ALL
SELECT id, word10 FROM word_table);

# remove the row contain no words
DELETE FROM word_list WHERE word1 IS NULL;

# however, there can be duplicated rows in the table. We can better
understand our data by including a primary key
# create a new column and make the column auto_increment.
ALTER TABLE `word_list`
ADD COLUMN `New_id` INT NOT NULL AUTO_INCREMENT AFTER `word1`,
           ADD PRIMARY KEY (`New_id`);


# detect word emotion
SELECT * FROM word_list a JOIN positive_emotion b WHERE  a.word1 LIKE
CONCAT('%',b.words,'%');

# Key takeway: you can use CONCAT to create new function
# you may notice that a.word1 could be 'larger' than b.words, such as the
word "recommended" in a.word1 matches the word "commend" in b.words
# What you could do to improve the command?

SELECT * FROM word_list a JOIN positive_emotion b WHERE CONCAT(' ',a.word1, '
') LIKE CONCAT('% ',b.words,' %');
CREATE TABLE temp AS (SELECT * FROM word_list a JOIN positive_emotion b WHERE
CONCAT(' ',a.word1, ' ') LIKE CONCAT('% ',b.words,' %'));

#add a new column to save emotion score.
ALTER TABLE `word_list` ADD COLUMN `Positive_emotion_score` TINYINT;

UPDATE word_list SET Positive_emotion_score = 0;
UPDATE word_list SET Positive_emotion_score = 1 WHERE New_id IN (SELECT
New_id FROM temp);
DROP table temp;

# Can you compute the positive emotion scores of each review title?

SELECT id, SUM(Positive_emotion_score) FROM word_list GROUP BY id;

#another way of presenting the results:
SELECT * FROM textmining_data tb1
JOIN (SELECT id, SUM(Positive_emotion_score) FROM word_list GROUP BY id) AS
tb2
ON tb1.id = tb2.id
```

```
# you can do the same to compute the negative emotion score.

# perform association analysis
SELECT  a.word1 as word1, b.word1 as word2, COUNT(*) as Frequency
FROM word_list as a
JOIN word_list as b
ON a.id = b.id AND a.word1 > b.word1
GROUP BY a.word1, b.word1;

CREATE TABLE association_table AS (SELECT  a.word1 as word1, b.word1 as
word2, COUNT(*) as Frequency
FROM word_list as a
JOIN word_list as b
ON a.id = b.id AND a.word1 > b.word1
GROUP BY a.word1, b.word1);

# let's see how customer evaluate e.g. breakfast or bathroom

SELECT * FROM association_table WHERE word2 LIKE 'breakfast' ORDER BY
frequency desc ;

SELECT * FROM association_table WHERE word2 LIKE 'bathroom' ORDER BY
frequency desc ;
```