



The lecture will start soon at
08:30



Aalto University
School of Business

MySQL for Data Analytics

Lecturer: Yong Liu

Contact me at: Yong.liu@aalto.fi

An Excellent 'Dictionary' of Basic MySQL Commands

- **MySQL by Examples for Beginners**

https://www3.ntu.edu.sg/home/ehchua/programming/sql/MySQL_Beginner.html

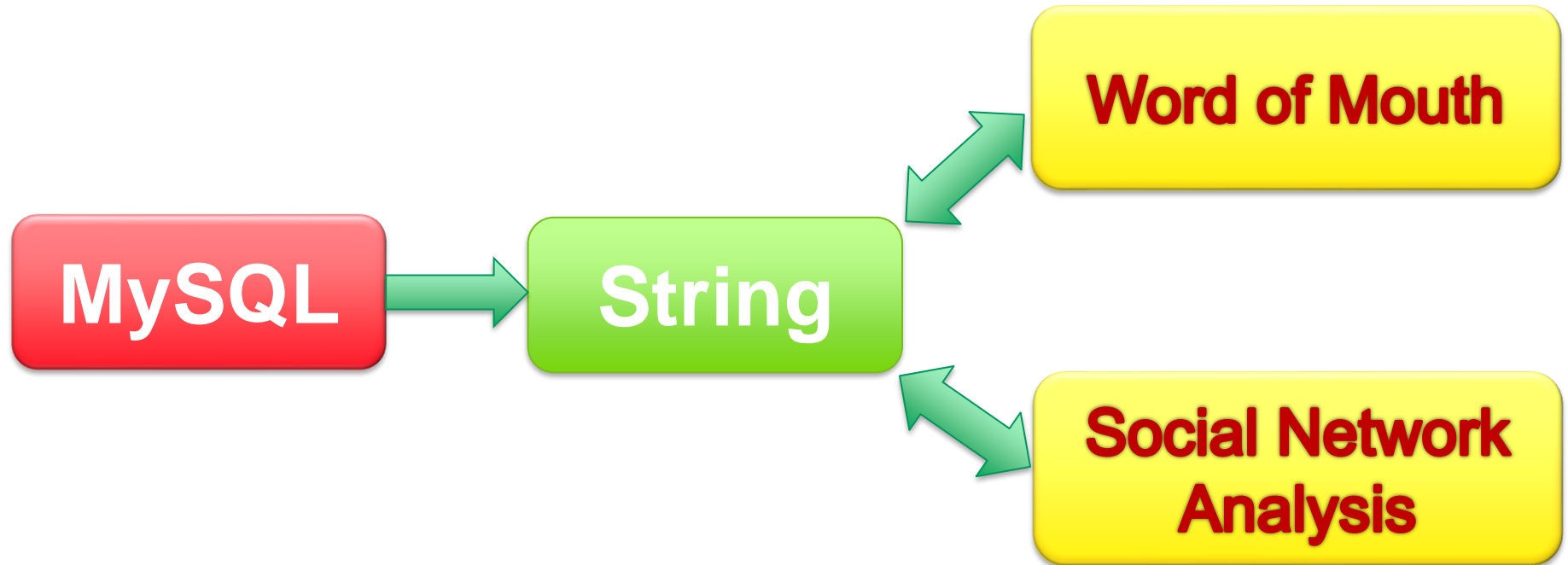
Objectives for class 4

- **String Analytics**
- **MySQL Keyword: Select (2)**
 - **Operation on string**
 - **Group by function (basics)**

‘String Analytics’

- **Not well handled by existing BI tools.**
- **An important component of tasks for many BI roles.**

MySQL for String Manipulation



A case of Supercell Game

- **Who is the main competitor?**

Word of mouth

“Excellent place in Helsinki”


★★★★★ Reviewed February 7, 2015

Really nice place in the heart of Helsinki. Excellent food and wines combined with passionate service. I can warmly recommend this place.

Was this review helpful? 1



“Unique dinning experience”

★★★★★ Reviewed August 13, 2013  via mobile

Enjoyed all the dishes. Beatifully presented and tasted good. Service was very attentive. I didn't try the drink menu but 8 course meal was a unique dinning experience.

Was this review helpful?



‘String’ to strategic insights



- **Yong Liu, Thorsten Teichert, Feng Hu, Hongxiu Li (2016) How do tourists evaluate Chinese hotels at different cities? Mining online tourist reviewers for new insights.**



Most often appeared words

Table 2. The hotel attributes that perform poorly in tourists' evaluation

| Shanghai (n =3298) | Sanya (n = 347) | Hangzhou (n = 451) | Guangzhou (n = 1172) | Beijing (n = 3752) |
|--------------------|------------------|--------------------|----------------------|--------------------|
| Service (12.5 %) | Service (13.5 %) | Service (9.0 %) | Service (9.3 %) | Service (9.9 %) |
| Location (6.6 %) | Resort (4.3 %) | Location (5.3 %) | Location (4.6 %) | Room (6.5 %) |
| Room (5.4 %) | Staff (2.5 %) | Room (5.0 %) | Room (4 %) | Location (6.0 %) |
| Staff (4.0 %) | Food (2.0 %) | Staff (3.7 %) | Dirty (2.7 %) | Staff (3.6 %) |
| Dirty (1.5 %) | Room (2.0 %) | Air (1.7 %) | Staff (2.6 %) | Dirty (2.5 %) |

Yong Liu, Thorsten Teichert, Feng Hu, Hongxiu Li (2016)

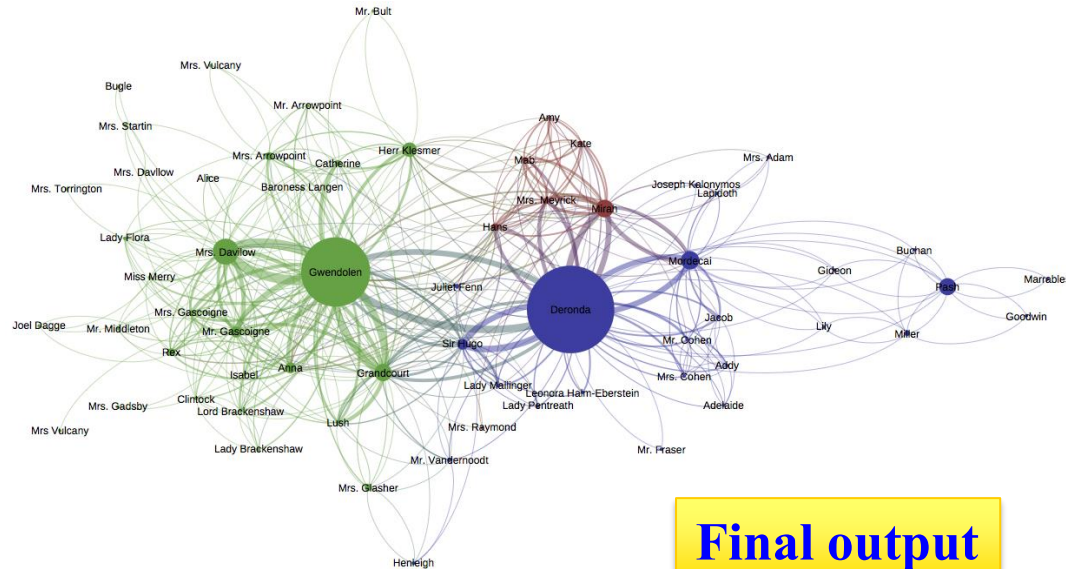
How do tourists evaluate Chinese hotels at different cities? Mining online tourist reviewers for new insights.

Raw data

| Communication ID | From | To | time |
|------------------|-------|----------|----------------------|
| 1 | Alice | Ben | 2005-02-03, 16:30:20 |
| 2 | Alice | Ben | 2005-05-08, 12:15:10 |
| 3 | Alice | Danielle | 2005-03-09, 09:32:26 |
| 4 | Ben | Carl | ... |
| 5 | ... | ... | ... |

Handled data

| | Alice | Ben | Carl | Danielle | Edgar |
|----------|-------|-----|------|----------|-------|
| Alice | 0 | 25 | 17 | 28 | 15 |
| Ben | 25 | 0 | 42 | 3 | 10 |
| Carl | 17 | 42 | 0 | 45 | 32 |
| Danielle | 28 | 3 | 45 | 0 | 13 |
| Edgar | 15 | 10 | 32 | 13 | 0 |



Final output

Detecting personality using words

- **<http://www.psychometrics.cam.ac.uk/news/the-language-of-social-media>**
- **Tal Yarkoni (2010). Personality in 100,000 Words: A large-scale analysis of personality and word use among bloggers, J Res Pers. 2010 Jun 1; 44(3): 363–373.**



Datasets for hands-on session

- **CFPB Consumer Complaint Database**
 - **Consumer Financial Protection Bureau (CFPB) launched a public Consumer Complaint Database on credit cards, mortgages, private student loan etc.**

<http://www.consumerfinance.gov/newsroom/consumer-financial-protection-bureau-launches-consumer-complaint-database/>

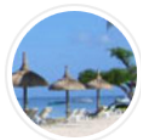
<https://www.consumerfinance.gov/data-research/consumer-complaints/>

| | Complaint ID | Product | Sub-product | Issue | Sub-issue | State | ZIP code | Submitted via | Date received | Date sent to company | Company | Company response | Timely response |
|---|--------------|-------------------------|----------------------------------|---|----------------------|-------|----------|---------------|---------------|----------------------|--------------------------------|-------------------------|-----------------|
| 1 | 1273083 | Bank account or service | (CD) Certificate of deposit | Account opening, closing, or management | | MA | 20902 | Phone | 03/09/2015 | 03/09/2015 | Ally Financial Inc. | In progress | Yes |
| 2 | 1272928 | Debt collection | Medical | Cont'd attempts collect debt not owed | Debt is not mine | IL | 61250 | Web | 03/09/2015 | 03/09/2015 | Transworld Systems Inc. | In progress | Yes |
| 3 | 1273115 | Bank account or service | Other bank product/service | Account opening, closing, or management | | PA | 19149 | Web | 03/09/2015 | 03/09/2015 | Santander Bank US | In progress | Yes |
| 4 | 1272530 | Debt collection | Other (phone, health club, etc.) | False statements or representation | Attempted to collect | MA | 02301 | Web | 03/08/2015 | 03/08/2015 | Ability Recovery Services, LLC | Closed with explanation | Yes |

Datasets for hands-on session

TripAdvisor data:

tripadvisor_data_for_handson_
assignment_ONLY.sql



AlanJWaters

Level 2 Contributor

7 reviews

3 helpful votes

“Excellent !”

NEW

Reviewed 3 days ago

Central position, clean, tidy, large with useful kitchen. Only slight draw back : we booked for a family, my wife and I and our two sons aged 21 and 14. We had a large comfy double bed but our sons had to share an uncomfortable double sofa bed which did not go down well !

Stayed January 2017, traveled with family

Sleep Quality

Rooms
Service

Helpful?

Thank AlanJWaters

Report

Ask AlanJWaters about Adina Apartment Hotel Berlin Checkpoint Charlie

This review is the subjective opinion of a TripAdvisor member and not of TripAdvisor LLC.

DRMCourse.tripadvisor_data_for_handson_assignment_ONLY: 1,979 rows total (approximately), limited to 1,000

Next Show all | Sorting Columns

| id | title | year_stayed | month_stayed | hotel_id | num_helpful_votes | review_date | via_mobile | rooms | value |
|-------------|--|-------------|--------------|-----------|-------------------|-------------|------------|-------|-------|
| 145,862,422 | "Good Place" | 2,012 | 9 | 2,145,772 | 0 | 2012-11-21 | FALSE | 0 | 0 |
| 145,862,517 | "Great Hotel With a Great Staff" | 2,012 | 8 | 98,821 | 0 | 2012-11-21 | FALSE | 0 | 0 |
| 145,862,957 | "Good quality budget hotel in expens..." | 2,012 | 11 | 81,042 | 1 | 2012-11-21 | FALSE | 3 | 3 |

Select...like [binary]...

| Complaint ID | Product | Sub_product | Issue |
|--------------|-------------|-------------|--|
| 2 151 | Credit card | | Billing disputes |
| 2 094 | Credit card | | Billing disputes |
| 2 418 | Credit card | | Rewards |
| 2 422 | Credit card | | Credit reporting |
| 2 046 | Credit card | | Credit card protection / Debt protection |
| 2 057 | Credit card | | Billing disputes |
| 2 066 | Credit card | | Billing disputes |
| 2 083 | Credit card | | Transaction issue |
| 2 121 | Credit card | | Unsolicited issuance of credit card |
| 2 199 | Credit card | | Transaction issue |
| 2 203 | Credit card | | Rewards |
| 2 259 | Credit card | | Other |
| 2 298 | Credit card | | Billing disputes |

How to retrieve records in which column ‘Issue’ starting with the word “Billing”?

Select...like [binary]...

- Keyword **Like** is used to compare strings.

- “%” represents a string of any possible length

`select * from customers where contactFirstName like 'ro%'`


- “_” represents a single possible character.

`select * from customers where contactFirstName like 'ro_'`

| customerNumber | customerName | contactLastName | contactFirstName | phone |
|----------------|------------------------------|-----------------|------------------|-------------------|
| 128 | Blauer See Auto, Co. | Keitel | Roland | +49 69 66 90 2555 |
| 452 | Mini Auto Werke | Mendel | Roland | 7675-3555 |
| 486 | Motor Mint Distributors Inc. | Salazar | Rosa | 2155559857 |

Select...like [binary]...

- Which query will return the result below?



Binary makes
comparison
case-sensitive

A. select * from customers where contactFirstName like **binary** 'ro%';

B. select * from customers where contactFirstName like **binary** 'Ro%';

| 🔑 customerNumber | customerName | contactLastName | contactFirstName | phone |
|------------------|------------------------------|-----------------|------------------|-------------------|
| 128 | Blauer See Auto, Co. | Keitel | Roland | +49 69 66 90 2555 |
| 452 | Mini Auto Werke | Mendel | Roland | 7675-3555 |
| 486 | Motor Mint Distributors Inc. | Salazar | Rosa | 2155559857 |

Questions

What does this query mean?

**Select * from customers where contactFirstName
not like 'ro__'**

**How to select customers whose first name
consists of 4 characters?**

select * from customers where contactFirstName like '_____'

Questions

- How to retrieve records in which ‘Issue’ **contains** the word ‘loan’?

select * from cfpb_complaints_2500 where _____

| Complaint ID | Product | Sub_product | Issue |
|--------------|-------------|--|---|
| 2 298 | Credit card | | Billing disputes |
| 2 315 | Mortgage | Conventional adjustable mortgage (A... | Loan servicing, payments, escrow acc... |
| 2 340 | Credit card | | Billing disputes |
| 2 348 | Credit card | | Billing disputes |

<http://presemo.aalto.fi/drm>

Select... IN...

- **Select ...**
[not] IN (element 1, element 2, ..., element N)

Select * from customers
where contactFirstName in ('Michael','Elizabeth')

Strings equal to "Michael" or "Elizabeth" will be returned

Select ... REGEXP...

- **MySQL REGEXP** performs a pattern match of a string expression against a pattern.
E.g., How to select the records in which addressline1 contains the words 'road', 'tower' or 'airport'

**Select * from customers
where addressLine1 regexp 'road|tower|airport'**

string contain 'road', 'tower' or 'airport' will be returned

Select ... REGEXP...(2)

| Function | Select * from <i>tableName</i> |
|---------------------|---|
| <code>^</code> | Where <i>ColumnName</i> REGEXP ' <code>^L</code> ' |
| <code>\$</code> | Where <i>ColumnName</i> REGEXP ' <code>c\$</code> ' |
| <code>.</code> | Where <i>ColumnName</i> REGEXP ' <code>^M..y\$</code> ' |
| <code>[]</code> | Where <i>ColumnName</i> REGEXP ' <code>[abc]</code> ' or ' <code>[0-9]</code> ' |
| <code>*</code> | Where <i>ColumnName</i> REGEXP ' <code>a*c</code> ' |
| <code>+</code> | Where <i>ColumnName</i> REGEXP ' <code>a+c</code> ' |
| <code>{N}</code> | Where <i>ColumnName</i> REGEXP ' <code>a{2}</code> ' |
| <code>{M, N}</code> | Where <i>ColumnName</i> REGEXP ' <code>a{1,2}</code> ' |

REGEXP is not case-sensitive!

<http://dev.mysql.com/doc/refman/5.7/en/regexp.html>

- **^L**: string start with L
- **c\$**: string ends with c
- **M..y\$**: string with L, end with y, and two characters in between.
- **'[abc]'**: string contain either a, or b or c.
- **'a*c'**: string in which character a appear before character c for 0 or several time.
- **'a+c'** string in which character a appear before character c for at least one time.
- **'a{2}'**: string in which 'aa' appears
- **'a{3}'**: string in which 'aaa' appears
- **'a{1,2}'**: string in which 'a' or 'aa' appears

What does the queries mean?

- **Select** Issue

from cfpb_complaints_2500 **where** Issue
regexp '^a|^b';

string starts with either letter 'a' or letter 'b'

- **select** Issue

from cfpb_complaints_2500 **where** Issue **not**
regexp '{1}';

string not contain any empty space

classicmodels.customers: 122 rows total (approximately)

| 🔑 customerNumber | customerName | contactLastName | contactFirstName | phone |
|------------------|--------------------|-----------------|------------------|-----------------|
| 216 | Enaco Distributors | Saavedra | Eduardo | (93) 203 4555 |
| 298 | Vida Sport, Ltd | Holz | Mihael | 0897-034555 |
| 344 | CAF Imports | Fernandez | Jesus | +34 913 728 555 |

- In the column 'phone' of table 'customers', a few phone numbers are found to be invalid because letters are included. Please return valid phone numbers that do not contain **ANY** letter [a - z].

Select * from customers where phone not regexp '[a-z]'

Select...Distinct...

- A product (**productCode**) can be purchased by many different customers, and thus appear multiple times in the table (**orderdetails**). Your boss wants to obtain the list of products (**productCode**) that have been purchased!

select distinct productCode from orderdetails

| 🔑 orderNumber | 🔑 productCode | quantityOrdered | priceEach | orderLineNumber |
|---------------|---------------|-----------------|-----------|-----------------|
| 10 100 | S18_1749 | 30 | 136 | 3 |
| 10 100 | S18_2248 | 50 | 55,09 | 2 |
| 10 100 | S18_4409 | 22 | 75,46 | 4 |
| 10 100 | S24_3969 | 49 | 35,29 | 1 |
| 10 101 | S18_2325 | 25 | 108,06 | 4 |
| 10 101 | S18_2795 | 26 | 167,06 | 1 |
| 10 101 | S24_1937 | 45 | 32,53 | 3 |

What does this query mean?

select distinct productCode, priceEach
from orderdetails

| 🔑 orderNumber | 🔑 productCode | quantityOrdered | priceEach | orderLineNumber |
|---------------|---------------|-----------------|-----------|-----------------|
| 10 100 | S18_1749 | 30 | 136 | 3 |
| 10 100 | S18_2248 | 50 | 55,09 | 2 |
| 10 100 | S18_4409 | 22 | 75,46 | 4 |
| 10 100 | S24_3969 | 49 | 35,29 | 1 |
| 10 101 | S18_2325 | 25 | 108,06 | 4 |
| 10 101 | S18_2795 | 26 | 167,06 | 1 |
| 10 101 | S24_1937 | 45 | 32,53 | 3 |
| 10 101 | S24_2022 | 46 | 44,35 | 2 |
| 10 102 | S18_1342 | 39 | 95,55 | 2 |
| 10 102 | S18_1367 | 41 | 43,13 | 1 |
| 10 103 | S10_1949 | 26 | 214,3 | 11 |
| 10 103 | S10_4962 | 42 | 119,67 | 4 |

| productCode | priceEach |
|-------------|-----------|
| S72_3212 | 43,68 |
| S72_3212 | 44,23 |
| S72_3212 | 44,77 |
| S72_3212 | 46,96 |
| S72_3212 | 47,5 |
| S72_3212 | 48,05 |
| S72_3212 | 48,59 |
| S72_3212 | 49,14 |
| S72_3212 | 51,32 |
| S72_3212 | 51,87 |
| S72_3212 | 52,42 |
| S72_3212 | 52,96 |
| S72_3212 | 53,51 |
| S72_3212 | 54,05 |
| S72_3212 | 54,6 |
| S72_1253 | 39,73 |
| S72_1253 | 40,22 |
| S72_1253 | 41,22 |
| S72_1253 | 41,71 |
| S72_1253 | 42,71 |
| S72_1253 | 43,2 |
| S72_1253 | 43,7 |
| S72_1253 | 44,2 |

Select...Group by...

IMPORTANT

- “**Group by**” is similar to “**Distinct**” that returns unique records based on the column(s) specified.
- **Select distinct productCode, priceEach from orderdetails**
- **Select productCode, priceEach from orderdetails group by productCode, priceEach**

The same results

Select...Group by...

- ‘Group by’ creates virtual sub-tables that share some common characteristics.

Select **orderNumber**, count(*)
from **orderdetails**
group by **orderNumber**

| orderNumber | productCode |
|-------------|-------------|
| 10,100 | S18_1749 |
| 10,100 | S18_2248 |
| 10,100 | S18_4409 |
| 10,100 | S24_3969 |
| 10,101 | S18_2325 |
| 10,101 | S18_2795 |
| 10,101 | S24_1937 |
| 10,101 | S24_2022 |
| 10,102 | S18_1342 |
| 10,102 | S18_1367 |
| 10,103 | S10_1949 |



Count
frequency

Select ... **ELT**...

- ELT() returns the *N*th element of the list of strings:

```
Select ELT(1, 'ej', 'Heja', 'hej', 'foo');
```

```
ELT(1, 'ej', 'Heja', 'hej', 'foo')  
ej
```

```
Select ELT(4, 'ej', 'Heja', 'hej', 'foo') as result;
```

```
result  
foo
```

Thinks about this query

| Name | Rank |
|-------|------|
| David | 3 |
| John | 1 |
| Lily | 2 |
| Adam | 4 |

Table `test`

```
SELECT `Name`, `Rank`,  
ELT(`Rank`, 'Ranks first',  
      'Ranks second',  
      'Ranks third',  
      'Ranks forth')  
as `rank`  
from test;
```


Result of the above query

| Name | Rank | rank |
|-------|------|--------------|
| David | 3 | Ranks third |
| John | 1 | Ranks first |
| Lily | 2 | Ranks second |
| Adam | 4 | Ranks forth |

Select ... **INSTR(str,substr)** ...

- Returns the position of the first occurrence of substring *substr* in string *str*.

```
mysql> SELECT INSTR('foobarbar', 'bar');
```

```
    -> 4
```

```
mysql> SELECT INSTR('xbar', 'foobar');
```

```
    -> 0
```

Select... **REPLACE**(*str,from_str,to_str*) ...

REPLACE() replaces all the occurrences of a substring within a string.

Select replace

('The food is good but the service is bad', 'but', '.')

as New_paragraph

New_paragraph

The food is good . the service is bad

What will be the result of following command?

```
select replace (  
replace('hello world', 'world', 'earth') , 'hello', 'hi')  
as funny_test
```

<http://presememo.aalto.fi/drm>

Select ... **LEFT**(*str,len*) ...

- Returns the leftmost *len* characters from the string *str*.

```
mysql> SELECT LEFT('foobarbar', 5);  
      -> 'fooba'
```

```
mysql> SELECT RIGHT('foobarbar', 4);  
      -> 'rbar'
```

Select ... **LENGTH**(*str*) ...

- **Possible applications: whether satisfied customers tend to speak more than those who are less satisfied?**

```
mysql> SELECT LENGTH('text');  
-> 4
```

```
select addressLine1, length(addressLine1) from customers
```

Select ... **Ltrim**(*str*) ...

- Returns the string *str* with leading space characters removed.

```
mysql> SELECT LTRIM('  barbar');  
-> 'barbar'
```

```
mysql> SELECT RTRIM 'barbar  ' ;  
-> 'barbar'
```

Question!

- In the table '**customers**', please find the customers whose First Name contains five characters after removing empty space from the right side.

| 🔑 customerNumber | customerName | contactLastName | ▲ contactFirstName |
|------------------|---------------------------|-----------------|--------------------|
| 282 | Souveniers And Things Co. | Huxley | Adrian |
| 398 | Tokyo Collectables, Ltd | Shimamura | Akiko |
| 237 | ANG Resellers | Camino | Alejandra |
| 443 | Feuer Online Stores, Inc | Feuer | Alexander |

Answer

- **select** contactFirstName **from** customers **where** contactFirstName **like** '_____';
- **select** contactFirstName **from** customers **where** rtrim(contactFirstName) **like** '_____';
- **select** contactFirstName **from** customers **where** length(rtrim(contactFirstName)) = 5



Five
underscores

Select ... LOWER(str) ...

- Returns the string **str** with all characters changed to lowercase.

```
mysql> SELECT LOWER('QUADRATICALLY') ;  
      -> 'quadratically'
```

- Many software treat strings in a case-sensitive manner. Thus, it is necessary to convert strings to lowercase letters before exporting data to these software for analysis.

Select ... **TRIM** ...

- **TRIM** ([**BOTH** | **LEADING** | **TRAILING** **FROM**] *str*)

```
mysql> SELECT TRIM('  bar ');
```

```
-> 'bar'
```

```
mysql> SELECT TRIM(LEADING 'x' FROM 'xxxbarxxx');
```

```
-> 'barxxx'
```

```
mysql> SELECT TRIM(BOTH 'x' FROM 'xxxbarxxx');
```

```
-> 'bar'
```

```
mysql> SELECT TRIM(TRAILING 'xyz' FROM 'barxxxyz');
```

```
-> 'barx'
```

Select ... SUBSTRING ...

```
mysql> SELECT SUBSTRING('Quadratically',5);  
      -> 'ratically'  
mysql> SELECT SUBSTRING('foobarbar' FROM 4);  
      -> 'barbar'  
mysql> SELECT SUBSTRING('Quadratically',5,6);  
      -> 'ratica'  
mysql> SELECT SUBSTRING('Sakila', -3);  
      -> 'ila'  
mysql> SELECT SUBSTRING('Sakila', -5, 3);  
      -> 'aki'  
mysql> SELECT SUBSTRING('Sakila' FROM -4 FOR 2);  
      -> 'ki'
```

SUBSTRING_INDEX(*str,delim,count*)

```
mysql> SELECT SUBSTRING_INDEX('www.mysql.com', '.', 2);  
      -> 'www.mysql'  
mysql> SELECT SUBSTRING_INDEX('www.mysql.com', '.', -2);  
      -> 'mysql.com'
```

| sent |
|--|
| first sentence. second sentence. third sentence. |
| first sentence. second sentence. third sentence. |

Table name: test; Column name: sent

SUBSTRING_INDEX(*str,delim,count*)

- **Returns the substring from string *str* before *count* occurrences of the delimiter *delim*. If *count* is positive, everything to the left of the final delimiter (counting from the left) is returned. If *count* is negative, everything to the right of the final delimiter (counting from the right) is returned.**
- **SUBSTRING_INDEX() performs a case-sensitive match when searching for *delim*.**

Help me with this challenge!


“Excellent place in Helsinki”

★★★★★ Reviewed February 7, 2015

Really nice place in the heart of Helsinki. Excellent food and wines combined with passionate service. I can warmly recommend this place.

Was this review helpful? 1

“Unique dinning experience”

★★★★★ Reviewed August 13, 2013  via mobile

Enjoyed all the dishes. Beatifully presented and tasted good. Service was very attentive. I didn't try the drink menu but 8 course meal was a unique dinning experience.

Was this review helpful?



Challenge Accepted

Challenge:

How to return the second sentence?

| sent |
|--|
| first sentence. second sentence. third sentence. |
| first sentence. second sentence. third sentence. |

Table name: test; Column name: sent

<http://presemio.aalto.fi/drm>

Hints

You can change * to
be some other
expression like
SUBSTRING_INDEX

- **create table** **new_table** **as select** * **from** **old_table**;

You can create a test table using the following comment

- **create table** **test**
(sent varchar(1000));
- **insert into** **test** **values**
('first sentence. second sentence. third sentence.');
- **insert into** **test** **values**
('first sentence. second sentence. third sentence.');

Solution I

- **create table** test2 **AS**
(select substring_index(sent, '.', 2) as first_two_sen from test);
- **Select** substring_index(first_two_sen, '.', -1) as second_sen **from** test2;

More sophisticated solution

```
select
substring_index(sent, '.', 1) as sen1,
substring_index(substring_index(sent, '.', 2), '.', -1)
as sen2,
substring_index(sent, '.', -2) as sen3
from test;
```

Challenge II

DRMCourse.tripadvisor_data_for_handson_assignment_ONLY: 1,981 rows total (approximately), limited to 1,000

| id | title | year_stayed | month_stayed | hotel_id | num_helpful_votes |
|-------------|---|-------------|--------------|-----------|-------------------|
| 145,862,422 | "Good Place" | 2,012 | 9 | 2,145,772 | 0 |
| 145,862,517 | "Great Hotel With a Great Staff" | 2,012 | 8 | 98,821 | 0 |
| 145,862,957 | "Good quality budget hotel in expensive city" | 2,012 | 11 | 81,042 | 1 |
| 145,863,456 | "Just book it!" | 2,012 | 11 | 113,329 | 5 |
| 145,863,801 | "InterContinental Mark Hopkins" | 2,012 | 11 | 115,623 | 0 |

Please calculate the number of words in the column 'title'!

Table name: test

<http://presemoo.aalto.fi/drm>



- **Select** title,
length(title) - length(replace(title, ' ', '')) + 1
as number_of_words
from test

Reflection: How to do this research?

Detecting personality using words

- <http://www.psychometrics.cam.ac.uk/news/the-language-of-social-media>
- Tal Yarkoni (2010). Personality in 100,000 Words: A large-scale analysis of personality and word use among bloggers, J Res Pers. 2010 Jun 1; 44(3): 363–373.

