- **Lab 1 – installing your SQL development environment**

Goal

The goal of this lab is to ensure that you are fully prepared to succeed in this class.

Submission

There is no submission. Student needs to have software installed and running to complete the course successfully. These instructions are Windows specific.

Tasks

1. Download Windows installer from MySQL: http://dev.mysql.com/downloads/installer/5.6.html. Make sure to select the full download installer (v5.6.20 as of this writting. Clocks in at ~248 MB). On the next screen click on "No thanks, just start my download" to avoid having to register.
2. Run the installer.
3. Select "Install MySQL Products"
4. Accept the license agreement and click next
5. Click Execute
6. Select "Custom" install type and click next.
7. Make sure only the following items are selected:
    a. MySQL Server 5.6.X with the following options:
        i. MySQL Server
        ii. Client Programs
        iii. Server data files
    b. Applications with the following options:
        i. My SQL Workbench CE
        ii. MySQL Utilities
    c. Deselect all MySQL connectors
    d. Documentation with all options selected
8. Click next and apply requirements if needed. Click next.
9. After install, click next
10. On the configuration screen accept defaults and click next
11. Create a root password. Make sure to take note of it because you will need to re-install if you lose it. You do not need to create any users. Click Next
12. Uncheck "Start the MySQL Server at System Startup. You don't want extra servers running in the background if you don't need them. Click Next.
13. Click Next.
14. Click Finish.
15. Launch MySQL Workbench if it is not already running.
16. Click on the "plus" icon next to MySQL Connections.
17. Give your server a name. Something like "MySQL Server local"
18. Next to the password field, click on "Store in Vault", enter the password you created earlier (step 11) and then click ok.
19. Click on "Test Connection". You should get a message that everything is ok.
20. Click on "Configure Server Management"
21. Click next, next, next, next and then continue.
22. Click OK.
23. Click on the newly created connection.
24. Click on "Startup/Shutdown" navigator item. Click "Stop Server" then click "Start Server" to make sure you have control of the MySQL Server.
25. Click on the "Home" icon on the top left of the tab bar.
26. Click Edit->Preferences menu.
27. Click SQL Queries tab and uncheck "Safe Updates" options.
28. Click on Model tab and make the following changes
    a. PK Column Name: id
    b. Column Type: VARCHAR(50)
29. Click on the Model:MySQL tab and make sure the following options are set:
    a. Default target MySQL Version: 5.5
    b. Default Storage Engine: MyISAM

Things to remember. In order to start you MySQL instance so that you can do your work, make sure to open the connection you created and then click on “Startup/Shutdown” navigator item. Click “Stop Server” then click “Start Server” to make sure you have control of the MySQL Server.

- 

Lab 2 – database design concepts applied

Goal

The goal of this lab is to start breaking in your database design skill set.

Submission

Lab is due by the start of next weeks lecture. Any submission past that time will receive an automatic 0.
Submit a hand drawn diagram with your name and student number at the top of the page.

Tasks

Design a 3 table database. The tables are as follows:
Customers – name, address, phone number and email

Products – product name, price/cost, description and on hand inventory

Customer Products – a HABTM table that links the Customers to the Products. Fields include references to the customer and product tables as well as quantity and dates.

Grading

18 points for the table definitions
2 points for naming conventions

20 points total

- 

Lab 3 - diagramming with MySQL Workbench part 1

Goal

The goal of this lab is to learn how to diagram database structures

Submission

Lab is due by the start of next weeks lecture. Any submission past that time will receive an automatic 0.
Submit the MySQL workbench diagram to blackboard named as follows: Last Name, first name – Lab 3.mwb

Task Summary

Design a 3 table database. The tables are as follows:
Customers – name, address, phone number and email

Products – product name, price/cost, description on hand inventory

Customer Products – a HABTM table that links the Customers to the Products. Fields include references to the customer and product tables as well as quantity and dates.

Detailed tasks

1. Open MySQL Workbench
2. click on File -> New Model
3. Double click on the database name (mydb) in the physical schema section of the interface.
4. Enter **sales** in the **name** field and press enter.
5. Click on “Add Diagram” icon in the model overview section of the interface. You should now be in design mode.
6. Click the add table icon in the toolbox on the left of the design area. Alternatively, press the **T** button. Click somewhere on your diagram.

7. Double click the newly created table object. And fill in the following properties:
    a. Table name is **customers**
    b. Primary key is called **id** and it is an INT with the following attributes. PK, NN, UN and AI.
    c. Create the following fields:
        i. name VARCHAR(50) NN
        ii. address VARCHAR(150)
        iii. city VARCHAR(50)
        iv. province VARCHAR(20)
        v. postal_code VARCHAR(10)
        vi. phone VARCHAR(14)
        vii. email VARCHAR(150)
8. Create a new table and double click the newly created table object. And fill in the following properties:
    a. Table name is **products**
    b. Primary key is called **id** and it is an INT with the following attributes. PK, NN, UN and AI.
    c. Create the following fields:
        i. name VARCHAR(50) NN
        ii. description VARCHAR(255)
        iii. price NUMERIC(6,2)
        iv. on_hand INT
9. Create a new table and double click the newly created table object. And fill in the following properties:
    a. Table name is **customer_products**
    b. Primary key is called **id** and it is an INT with the following attributes. PK, NN, UN and AI.
    c. Create the following relatons. To create a relation, click on the 1:n non-identifying relation icon or press the **2** key. Click on the child table, then the parent table.
        i. Child **customer_products**, parent **customers**
        ii. Child **customer_products**, parent products
    d. Rename the newly create foreign key fields so that they are singular.
        i. customer_id
        ii. product_id
    e. Create the following fields:
        i. quantity INT
        ii. purchase_date DATETIME

Grading

6 points for the table definitions
2 points for relations
2 points for naming conventions

10 points total

- 

Lab 4 – diagramming MySQL Workbench part 2

Goal

The goal of this lab diagram a database and push it to a database server. Also covered are how to forward engineer and how to synchronize changes.

Submission

Lab is due by the start of the next week's lecture. Any submission past that time will receive an automatic 0.
Submit a file containing the SQL commands to blackboard named as follows:

- **Last Name, first name – Lab 4.sql**

Task Summary

Design a 1 table database. Add a field, change a field and remove a field with the changes and synchronized to a database.
See Lab 3 for the mechanical steps to create a database diagram and a table. Make sure your MySQL server is up and running before doing this lab.

Detailed tasks

1. Create a diagram and name the database lab_4.
2. Create a table with the following properties
    a. Name: lab4_table
    b. Fields
        i. **id** INT PK NN UN AI
        ii. **name** VARCHAR(50)
3. Now forward engineer you design.
    a. Click on Database ->Forward Engineer
    b. Select the localhost connection as created in Lab 1. Click next.
    c. Do not change any settings on the following screen and click next.
    d. Ensure that **Export MySQL Table Objects** is checked and press next.
    e. Either save the content of the generated code to the clipboard or save it as a file. You will need this for your lab submission. Click Next and then Close.
4. Double click the **lab_4** table and add the following fields:
    a. email VARCHAR(150)
    b. phone VARCHAR(15)
5. Now synchronize your changes
    a. Click Database->Synchronize Model
    b. Select localhost and click Next. Click Next again.
    c. Make sure that lab_4 is selected and click Next and Next.
    d. Click on lab4_table and review the schema changes. Click Next.
    e. Either save the content of the generated code to the clipboard or save it as a file. You will need this for your lab submission. Click Execute and then Close.
6. Do the following tasks
    a. Rename the table to yourfirstname_lab4. i.e. dan_lab4
    b. Delete the **email** field from the diagram
    c. Rename **phone** to **telephone.**
7. Synchronize your changes like step 5 above.
8. Take the content of each of the files you created and paste them into a single file. This is the file you are to submit.

Grading

6 points. 1 for each task.

- 

Lab 5 – Importing/Exporting databases, generating test data

Goal

The goal of this lab is to learn how to import and export databases using MySQL Workbench and how to generate test data and add it to your database.

Submission

Lab is due by the start of the next week's lecture. Any submission past that time will receive an automatic 0.

Submit a file containing the SQL commands to blackboard named as follows:

- **Last Name, first name – Lab 5.sql**

Task Summary

Import a pre-created database. Generate some test data and import that into the database. Export the database.

Detailed tasks

1. Importing a pre-created database.
    a. Download lab5.sql that is attached to this lab
    b. Open up your local connection and make sure the server is running. Refer to lab 1 to see how to start it.
    c. In the Navigator, select Data Import/Restore.
    d. Select “Import from Self Contained File” and select lab5.sql that you download in step 1.
    e. Click start.
    f. In the Navigator click the refresh icon next to the Schemas heading.
    g. Double click lab5.
2. Generating test data. In this step, everything is case sensitive.
    a. Using your preferred web browser, go to: http://www.generatedata.com/
    b. Fill in the DATA SET form as follows:
        i. Column title: first_name, data type: Name, example “Alex (Any Gender)”
        ii. Column title: last_name, data type: Name, example “Smith (Surname)”
        iii. Column title: email, data type: Email
    c. In the EXPORT TYPES section, click the SQL tab. And enter the following options:
        i. Database table name: authors
        ii. Database type: MySQL
        iii. Uncheck “Include Create Table query” and “Include Drop Table Query”
        iv. Statement Type: Insert
        v. Primary key: None
    d. In the green box at the bottom:
        i. Click “prompt to download”.
        ii. Click generate
3. Importing the data.
    a. In MySQL workbench, click on the “open” icon in the SQL Editor toolbar. It looks like a small folder. Select the file you just downloaded from generatedata.com.
    b. Click the Run icon. It looks like a lightning bolt.
4. Redo steps 2 and 3 except this time the Database table name is “articles” and DATA SET should look like:
    a. Column title: name, data type: Random number of words, Generate 1 to 3 words”
    b. Column title: article_text, data type: Random number of words, Generate 250 to 1000 words”
    c. Column title: published, data type: number range, Between 0 and 1

d. Column title: category_id, data type: number range, Between 1 and 6
e. Column title: author_id, data type: number range, Between 1 and 100

5. Exporting the data.
   a. In the navigator click “Data Export”
   b. In the list of objects, check lab5.
   c. Make sure “Export to Self-Contained File” is selected and pick a path. Change the filename to Last Name, first name – Lab 5.sql
   d. Click Start Export.

Submit the export file to blackboard.

Grading

5 points. 1 for each numbered task.

Lab 6 – DDL and DML

Attached Files:

- technews.sql (16.234 MB)
- tech_news.png (50.471 KB)

Goal

The goal of this lab is to practice using DDL to create and alter some tables and using DML to add and query the data.

Submission

Lab is due by the start of the next week's lecture. Any submission past that time will receive an automatic 0.
Submit a file, containing the SQL commands for tasks 3-16, to blackboard named as follows:

- **Last Name, first name – Lab 6.sql**

Task Summary

Import a pre created database. Alter the database structure. Execute some basic queries.

Detailed tasks

1. Download **technews.sql** file that is attached to this lab.
2. Refer to task #1 from lab 5 for guidance on importing tech news database. In MySQL workbench SQL Editor, make sure to have **tech_news** database selected (double click).
3. Create a table with the following properties
   a. Table name: comment_statuses
   b. Fields defined as follows:
      i. id INT UNSIGNED PRIMARY KEY AUTO_INCREMENT
      ii. name VARCHAR(50)
4. Insert into following values into the **comment_statuses** table
   a. Waiting for approval
   b. Approved
   c. Blocked
5. Alter **comments** table to have the following field:
   a. comment_status_id INT UNSIGNED
6. update all **comments** to have a **comment_status_id** of 1;
7. select all **users**
8. select user with id of 233;
9. select all **users** that have an id between 25 and 75 ;

10. select all user_**types**
11. using the information you found in 10, select all **users** that are **editors**;
12. using the information you found in 10, select all **users** that are either **editors** or **authors**;
13. select all **comments** that have a **created** date before 2006.
14. select all **comments** that have a **created** date later than Jan 1, 2006 but before Jan 1,2007.
15. Update all **comments** that were **created** on or after Jan 1, 2008 to have a **comment_status_id** of 'Approved';
16. Delete all **comments** that were **created** before 2007.

Grading

14 points. 1 for each numbered task from 3 to 16.

3)CREATE TABLE comment_statuses(
id INT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
name VARCHAR(50)
);
4)INSERT INTO comment_statuses
(name)
values
('Waiting for approval'), ('Approved'), ('Blocked');
5)ALTER TABLE comments ADD COLUMN comment_status_id INT UNSIGNED;
6)UPDATE comments SET comment_status_id=1;
7)SELECT * FROM users;
8)SELECT * FROM users WHERE id=233;
9) SELECT * FROM users WHERE id>25 AND id<75;
10)SELECT * FROM user_types;
11)SELECT * FROM users WHERE user_type_id=2;
12)SELECT * FROM users WHERE user_type_id=2 OR user_type_id=3;
13)SELECT * FROM comments WHERE created < '2006';
14)SELECT * FROM comments WHERE created > '2006-01-01' AND created<'2007-01-01';
15)UPDATE comments SET comment_status_id=2 WHERE created >='2008-01-01';
16)DELETE FROM comments WHERE created <'2007';

- 

Lab 7 – Aggregates, Aliases, Subqueries and JOINS

Goal

The goal of this lab is to practice using aggregate functions and JOINS.

Submission

Lab is due by the start of the next week's lecture. Any submission past that time will receive an automatic 0.
Submit a file containing the SQL commands to blackboard named as follows:

- **Last Name, first name – Lab 7.sql**

Task Summary

Using a variety of aggregate functions and joins, you will data mine the tech_news database for information. Uses the tech_news database imported in Lab 6.

Detailed tasks

1. Count how many **users** there are.
2. Alias the results of the count from task 1 to have the name of "Users"
3. Count how many users where created before 2010
4. Count **articles** grouped by who created them (**created_by_id**)
5. Count **articles** grouped by the **name**, aliased as Author, of who created them (**created_by_id**)
6. Count how many **comments** per **article**. Return the **name** of the article and **comment** count aliased as Comments.

7. List all the articles **id**, **name** and their **state**, not the **state** id. Alias article **id** as Article Number, article **name** as Article and state **name** as State.
8. Redo # 7, this time restricting only to states of New and Waiting for Approval.
9. Redo # 8, this time use a subquery
10. What is the average, minimum and maximum number of articles per author?

Grading

10 points. 1 for each numbered task. Hint for #10, http://lmgtfy.com/?q=mysql+avg+count

Question 1
```
SELECT COUNT(*)
FROM users
```

Question 2
```
SELECT COUNT(*) AS 'Users'
FROM users
```

Question 3
```
SELECT COUNT(*) AS 'Users'
FROM users
WHERE created < '2010%'
```

Question 4
```
SELECT created_by_user_id AS 'Created By User ID',
COUNT(*) AS 'Number of Articles'
FROM articles
GROUP BY created_by_user_id
```

Question 5
```
SELECT created_by_user_id AS 'Author',
COUNT(*) AS 'Number of Articles'
FROM articles
GROUP BY created_by_user_id
```

Question 6
```
SELECT articles.name AS 'Articles',
COUNT(*) AS 'Comments'
FROM comments
JOIN articles
ON (articles.id=comments.article_id)
GROUP BY articles.name
ORDER BY articles.name
```

Question 7
```
SELECT
articles.id AS 'Article Number',
articles.name AS 'Article',
article_states.name AS 'State'
FROM articles
JOIN article_states
ON (article_states.id= articles.article_state_id)
GROUP by articles.id
ORDER BY articles.id
```

Question 8
```
SELECT
articles.id AS 'Article Number',
articles.name AS 'Article',
article_states.name AS 'State'
```

```
FROM articles
JOIN article_states
ON (article_states.id= articles.article_state_id)
WHERE article_states.name = 'New'
OR article_states.name = 'Waiting for Approval'
GROUP by articles.id
ORDER by articles.id
```

Question 9

```
SELECT
articles.id AS 'Article Number',
articles.name AS 'Article',
FROM articles
        (SELECT article_states.name AS 'State'
        WHERE article_states.name = 'New'
        OR article_states.name = 'Waiting for Approval')
GROUP by articles.id
ORDER by articles.id
```

Question 10

```
SELECT max(articles.id),
      min(articles.id),
      avg(articles.id)
FROM
(SELECT created_by_user_id AS 'Created By User ID',
COUNT(*) AS 'Number of Articles'
FROM articles
```

- 

Lab 8 – Left/Right joins and using Views

Goal

The goal of this lab is to practice using left/right joins and creating, altering and using views.

Submission

Lab is due by the start of the next week's lecture. Any submission past that time will receive an automatic 0.
Submit a file containing the SQL commands to blackboard named as follows:

- **Last Name, first name – Lab 8.sql**

Task Summary

Using the tech_news database, create and use left and right views. Create, use and alter several views.

Detailed tasks

1. Create and execute an SQL statement that retrieves:
    a. the name of the user aliased as ‘author’
    b. the name of the user_type associated with the author aliased as ‘type’
    c. the name of the articles created by the user.
2. Redo #1, but this time, left join the articles. Make note of what is difference
3. Truncate the table temp_users. Make note of the response from the server.

4. Execute the following command:
   insert into temp_users select id from articles order by rand() limit 250;
   Make note of the response from the server.
5. Execute the following command:
   delete from articles where id in (select id from temp_users);
   Make note of the response from the server.
6. Redo #1, but this time right join the articles. What do you notice?
7. Execute the following statement:
   update articles set created_by_user_id = 0 where created_by_user_id in (select id from temp_users);
8. Redo #6, this time with the addition of ordering by 'author'.
9. Create a view of #1. Name it authors_with_articles.
10. Create a view of #1, this time count the articles, aliased as 'article_count' and grouped by 'author', instead of outputting the name of the article. Name it author_article_count.
11. Select all records from author_article_count.
12. Select the average number of articles from author_article_count.
13. Redo #12, this time group by 'type'. Submit the averages.

Grading

15 points. 1 for each numbered task.

Task 1

```
SELECT users.name AS 'author',
user_types.name AS 'type',
articles.name AS 'article name'
FROM users
JOIN
 user_types ON (user_types.id = users.user_type_id)
JOIN articles ON (users.id = articles.created_by_user_id)
GROUP BY users.name, articles.name
ORDER BY users.name
```

Task 2

```
SELECT
users.name AS 'author',
user_types.name AS 'type',
articles.name AS 'article name'
FROM articles
LEFT JOIN
users ON (users.id = articles.created_by_user_id)
JOIN
user_types ON (user_types.id = users.user_type_id)
GROUP BY users.name, articles.name
ORDER BY articles.name
```

Task 3

```
15:57:51        truncate table temp_users    0 row(s) affected   0.016 sec
```

Task 4

```
16:02:11        insert into temp_users select id from articles order by rand() limit 250        250 row(s) affected Records: 250
Duplicates: 0  Warnings: 0          0.015 sec
```

Task 5

```
16:03:01        delete from articles where id in (select id from temp_users)          250 row(s) affected          0.813 sec
```

Task 6

```
SELECT
```

```
users.name AS 'author',
user_types.name AS 'type',
articles.name AS 'article name'
FROM articles
RIGHT JOIN
users ON (users.id = articles.created_by_user_id)
JOIN
user_types ON (user_types.id = users.user_type_id)
GROUP BY users.name
ORDER BY articles.name
```

Many of the article names are null

Task 7

```
UPDATE articles SET created_by_user_id = 0
WHERE created_by_user_id IN (SELECT id FROM temp_users);
```

16:05:46 update articles set created_by_user_id = 0 where created_by_user_id in (select id from temp_users) 195 row(s) affected Rows matched: 195 Changed: 195 Warnings: 0 0.860 sec

Task 8

```
SELECT
users.name AS 'author',
user_types.name AS 'type',
articles.name AS 'article name'
FROM articles
RIGHT JOIN
 users ON (users.id = articles.created_by_user_id)
JOIN
 user_types ON (user_types.id = users.user_type_id)
GROUP BY users.name
ORDER BY users.name
```

Task 9

```
CREATE VIEW authors_with_articles AS
SELECT users.name AS 'author',
user_types.name AS 'type',
articles.name AS 'article name'
FROM users
JOIN
 user_types ON (user_types.id = users.user_type_id)
JOIN articles ON (users.id = articles.created_by_user_id)
GROUP BY users.name, articles.name
ORDER BY users.name
```

Task 10

```
CREATE VIEW author_article_count AS
SELECT COUNT(article) AS 'article_count',
users.name AS 'author',
user_types.name AS 'type',
articles.name AS 'article name'
FROM users
JOIN
 user_types ON (user_types.id = users.user_type_id)
JOIN articles ON (users.id = articles.created_by_user_id)
```

GROUP BY 'author'

Task 11
SELECT * FROM author_article_count

Task 12
SELECT AVG('article name') FROM author_article_count

Task 13
SELECT AVG('article name') FROM author_article_count
GROUP BY 'type'
0 returned

- 

Lab 9 – Unions, Intersects and Excepts

Attached Files:

- lab9_products.sql (19.011 KB)

Goal
The goal of this lab is to practice using Unions, Intersects and Excepts.

Submission
Lab is due by the start of the next week's lecture. Any submission past that time will receive an automatic 0.
Submit a file containing the SQL commands to blackboard named as follows:

- **Last Name, first name – Lab 9.sql**

Task Summary
Create a series of Unions, Intersects and Excepts.

Detailed tasks
1. Import lab9_products.sql file. Refer to Lab 5 on how to import a database.
2. Build a list of all products (product names only) from both tables
3. Build a list of all products in products1 that is not in products2
4. Build a list of all products in products2 that is not in products1
5. Build a list of all products with their versions from both tables
6. Concatenate product names and product versions from both tables.

Grading
5 points. 1 for each task from 2 to 6.

Task 2
SELECT product FROM products1
UNION
SELECT product_name FROM products2

Task 3

```
SELECT product FROM products1
WHERE product NOT IN
(
SELECT product_name FROM products2)
```

Task 4

```
SELECT product_name FROM products2
WHERE product_name NOT IN
(
SELECT product FROM products1)
```

Task 5

```
SELECT product as 'Products',
version as 'Version'
FROM products1
UNION
SELECT product_name as 'Products',
version_number as 'Version'
FROM products2
```

Task 6

```
SELECT CONCAT(product, version)
FROM products1
UNION
SELECT CONCAT(product_name, version_number)
FROM products2
```

- 

Lab 10 – User defined functions

Goal

The goal of this lab is to practice creating, using, altering and deleting user defined functions.

Submission

Lab is due by the start of the next week's lecture. Any submission past that time will receive an automatic 0.
Submit a file containing the SQL commands to blackboard named as follows:

- **Last Name, first name – Lab 10.sql**

Include delimiter statements when submitting the lab. Make a backup of your database before doing this lab.

Task Summary

Using the tech_news database, create a function to clean up and make the article names URL safe.

Detailed tasks

1) Alter the table articles by adding the following column:
    a. urlsafe VARCHAR(50)
2) Create a user defined function, named clean_string, to strip out spaces. The user defined function will accept a string(varchar) as an argument. (Just like a function declaration in Java or PHP)
3) Test your function by running something like:
    a. select clean_string(" this is a test");
4) Now test your function by running a string through that has other characters present such as a period of a semi colon.
5) Alter your function to strip out the following characters in addition to spaces:
   .;!@#$%*
6) Now run your function against the name field of articles.
7) Alter your function to clean out any additional special characters that showed up in #6.
8) Update articles table by setting urlsafe field to the result of the clean_string function.

Grading

8 points. Hint for how to replace a string: http://lmgtfy.com/?q=How+to+replace+a+string+in+MySQL+query+stack+overflow

Task 1:

```
ALTER TABLE articles ADD COLUMN  urlsafe VARCHAR(50);
```

Task 2:

```
DELIMITER $$
CREATE FUNCTION clean_string ( originalString VARCHAR(50) )
RETURNS VARCHAR(50)
BEGIN
DECLARE result VARCHAR(50);
SET result = REPLACE( originalString, '  ', ' ' );
WHILE (result <> originalString) DO
SET originalString = result;
SET result = REPLACE( originalString, '  ', ' ' );
END WHILE;
RETURN result;
END$$
DELIMITER ;
```

Task 3

```
SELECT clean_string(" this is a test");
```

Task 4

```
select clean_string(" this is a.:Gs.; test");
```

Task 5

```
DELIMITER $$
CREATE FUNCTION clean_string ( originalString VARCHAR(50) )
RETURNS VARCHAR(50)
BEGIN
DECLARE result VARCHAR(50);
SET result = REPLACE( originalString, ' ', " );
WHILE (result <> originalString) DO
SET originalString = result;
SET result = REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE
( originalString, ' ', "),  '.', "), ';', "), '!', "), '@', "), '#', "), '$', "), '%', "), '*', " );
END WHILE;
RETURN result;
END$$
DELIMITER ;
```

Task 6

```
SELECT clean_string (name) FROM articles
```

Task 7

```
DELIMITER $$
CREATE FUNCTION clean_string ( originalString VARCHAR(50) )
RETURNS VARCHAR(50)
BEGIN
DECLARE result VARCHAR(50);
SET result = REPLACE( originalString, ' ', '' );
WHILE (result <> originalString) DO
SET originalString = result;
SET result = REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE
( originalString, ' ', ''),  '.', ''), ';', ''), '!', ''), '@', ''), '#', ''), '$', ''), '%', ''), '*', ''), ',', '' );
END WHILE;
RETURN result;
END$$
DELIMITER ;
```

Task 8

```
UPDATE articles SET urlsafe = clean_string (name)
```

- 

Lab 11 – Triggers

Goal

Goal of this lab is to learn how to create and user triggers

Submission

Lab is due by the start of the next week's lab. Any submission past that time will receive an automatic 0.
Submit a file containing the SQL commands to blackboard named as follows:

- **Last Name, first name – Lab 11.sql**

Include delimiter statements when submitting the lab.

Task Summary

Create a trigger that logs changes to the users table in the tech_news database.

Detailed tasks

1. Create a user_log table with the following fields.
    a. id INT(10) UNSIGNED PRIMARY KEY auto_increment
    b. user_id INT (10)
    c. name VARCHAR (50)
    d. username VARCHAR (32)
    e. password VARCHAR (32)
    f. email VARCHAR(255) NOT NULL
    g. user_type_id INT (11)
    h. created DATETIME NOT NULL

2. Create a trigger that will log all changes to the users table into the user_log table. The trigger should insert the OLD values into the log table.
3. Update user with id 1 to have an user_type_id of 2.
4. Select everything from user_log

Grading

4 points. 1 for each numbered task.

Task 1:

```
CREATE TABLE user_log (
  id INT (10) UNSIGNED PRIMARY KEY AUTO_INCREMENT,
  user_id INT (10),
  name VARCHAR (50),
  username VARCHAR (32),
  password VARCHAR (32),
  email VARCHAR (255) NOT NULL,
  user_type_id INT (11),
  created DATETIME NOT NULL
);
```

Task 2:

```
drop trigger update_user;
DELIMITER $$
CREATE TRIGGER update_user AFTER UPDATE on users
FOR EACH ROW
BEGIN
INSERT INTO user_log
(user_id, name, username, password, email, user_type_id, created)
VALUES (OLD.id, OLD.name, OLD.username, OLD.password, OLD.email, OLD.user_type_id, NOW());
END$$
DELIMITER ;
```

Task 3

```
UPDATE users SET user_type_id = 2 WHERE id = 1;
```

Task 4

```
SELECT * FROM user_log;
```

- 

Lab 12 – Specialty queries

Attached Files:

- lab12.sql (4.026 KB)

Goal

Goal of this lab is to learn how to use specialty queries such as finding duplicate rows and adding data to a table using a select statement.

Submission

Lab is due by the start of the next week's lecture. Any submission past that time will receive an automatic 0.
Submit a file containing the SQL commands to blackboard named as follows:

- **Last Name, first name – Lab 12.sql**

Make a backup of your database before doing this lab.

Task Summary

Using a variety of queries, you will clean up a table called drugs in the lab12 database.

Detailed tasks

1) Import lab12.sql
2) Select all from drugs table.
3) Identify all the duplicate drugs
4) Create a temporary table to help identify the duplicate rows
5) Insert the rows to identify the duplicate entries into the temporary table
6) Check to make sure that the data has been added to the temporary table
7) Delete the duplicate rows.
8) Check to see if all the duplicate rows have been cleaned up by redoing #3.

Grading

7 points. 1 for each numbered task from 2 to 8.

Task 2
```
SELECT * FROM drugs;
```

Task 3
```
SELECT name, count(*)
FROM drugs
GROUP BY name
HAVING count(*) > 1;
```

Task 4
```
CREATE TEMPORARY TABLE duplicate_drugs(
min_id INT UNSIGNED NOT NULL,
name VARCHAR(100) DEFAULT NULL
);
```

Task 5
```
INSERT INTO duplicate_drugs (min_id, name)
(SELECT MIN(id), name FROM drugs GROUP BY name HAVING count(*) > 1);
```

Task 6
```
SELECT * FROM duplicate_drugs;
```

Task 7
```
DELETE FROM drugs
WHERE EXISTS(
SELECT * FROM duplicate_drugs
WHERE duplicate_drugs.name = drugs.name and duplicate_drugs.min_id <> drugs.id
 );
```

Task 8
```
SELECT name, count(*) FROM drugs GROUP BY name HAVING count(*) > 1;
```