

COMP40725

Lab Book 2: Learning SQL using The Supplier Parts Database

NB Using the shell provided in XAMPP has to added benefits

1. MySQL is automatically in the path meaning that you do not need to change to the insall directory to log on
2. The history of your commands is stored and may be accessed by using the “Up Arrow” key

PART 1: Setup the lab and load SupplierPartsDB database

1. Create a folder called *Lab2 in your COMP40725*. You may wish to have this inside a folder that is being synced by Google Drive
2. Download the file called *Lab_Book_2_supplierParts.sql* from Moodle into the Lab2 folder.
3. Open the above SQL file in a text editor which provides highlighting to see the list of SQL queries that are contained in the file e.g.
 - Windows: Notepad++ <http://notepad-plus-plus.org/download/v6.5.3.html> or
 - Mac: TextWrangler <http://www.barebones.com/products/textwrangler/download.html>
4. Load XAPP
5. Start your MySQL Server
6. Open a shell/ terminal
 - a. Start the XAMP shell (MySQL will be in your path) or
 - b. Load a windows shell (MySQL may not be in your path) by pressing “Windows Key” + “R” and then typing “cmd” and then hitting return or
 - c. In Mac / Linux open a terminal (MySQL may not be in your path)
7. Load the database into MySQL using one of the below options (**blue indicates prompts**):

From the XAMPP shell

1. MySQL is in your path and the SQL file is in your present working directory:

```
# mysql -u root -p
mysql> source Lab_Book_2_supplierParts.sql
```
2. MySQL is in your path and the SQL file is NOT in your present working directory:

```
# mysql -u root -p
mysql> source D:\GoogleDrive.UCD\COMP40725\
\Lab2\Lab_Book_2_supplierParts.sql
```

This is the best option

From cmd.exe

3. MySQL is in your path and the SQL file is in your present working directory:

```
c:\xampp\mysql\bin>mysql> mysql -u root -p <
Lab_Book_2_supplierParts.sql
```
4. MySQL is in your path and the SQL file is NOT in your present working directory:

```
c:\xampp\mysql\bin>mysql> mysql -u root -p <
D:\GoogleDrive.UCD\COMP40725\
\Lab2\Lab_Book_2_supplierParts.sql
```

Note: If anything goes wrong with the database (accidental deletion, etc.), you can repeat this step to restore the database to its original state.

1. If the import was successful, you should **not** see any error messages. To check that the import was successful, make sure you are logged in to mysql and execute

`SHOW DATABASES;`

You should see a new database called *SupplierPartsDB*. If so, execute:

`USE SupplierPartsDB;`

2. Execute:

`SHOW TABLES;`

```
mysql> SHOW TABLES;
+-----+
| Tables_in_supplierpartsdb |
+-----+
| Parts                      |
| Supplier                  |
| Supplies                  |
+-----+
3 rows in set (0.00 sec)
```

This should return the list of tables called: *Supplier*, *Parts* and *Supplies*. As can be seen above.

Execute the following command for each table to see its structure:

`DESCRIBE tablename;`

PART 2: Lab Questions

Your submission should be named lab2_studentNumber_firstName_LastName.doc

Open Notepad++ or TextWrangler and create and save an empty file called tmp.sql in your Lab 2. You can write your statements in this file before you execute them to assist you in designing them.

Support for this practical can be found in the following ways

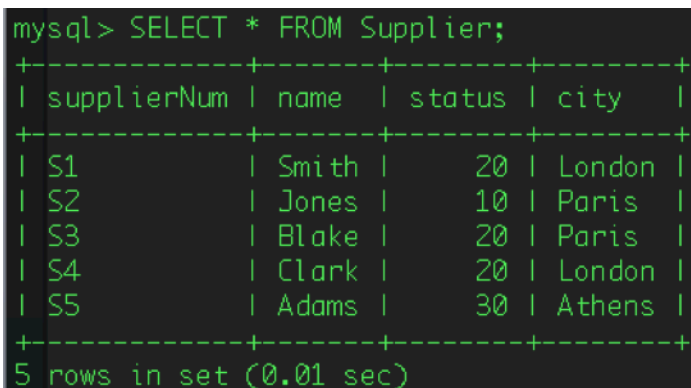
1. Lecture Notes.
2. Google “MySQL name_of_command” and you will find some examples.
3. Demonstrator – you can ask us any question no matter how basic.

For each of the questions below, include a screenshot of your SQL query being executed and the resultant changes to the table (where necessary).

Q1. The SQL CREATE TABLE <tableName> AS... can be used to create a table from an existing table in MySQL. Create a table named ‘SupplierA’ built from the data contained in the Supplier table for supplierNum S1 and S2. You should provide a screenshot of both the CREATE TABLE query and the result of the query: SELECT * FROM SupplierA. Part of the solution is in the slides!

1. To provide a basis for comparison, we should show the data contained in the table, Supplier, from which the table, SupplierA, will be built:

`SELECT * FROM Supplier;`



```
mysql> SELECT * FROM Supplier;
+-----+-----+-----+-----+
| supplierNum | name  | status | city  |
+-----+-----+-----+-----+
| S1          | Smith | 20     | London |
| S2          | Jones | 10     | Paris  |
| S3          | Blake | 20     | Paris  |
| S4          | Clark | 20     | London |
| S5          | Adams | 30     | Athens |
+-----+-----+-----+-----+
5 rows in set (0.01 sec)
```

2. Now, let's create the new custom table using the following query:

```
CREATE TABLE SupplierA AS SELECT supplierNum, name, status, city
FROM Supplier WHERE supplierNum='S1' OR supplierNum='S2';
```

```
mysql> CREATE TABLE SupplierA AS SELECT supplierNum, name, status, city FROM Supplier WHERE supplierNum='S1' OR
supplierNum='S2';
Query OK, 2 rows affected (0.12 sec)
Records: 2 Duplicates: 0 Warnings: 0
```

3. The result of creating table, SupplierA, using the 'SELECT' statement:

SELECT * FROM SupplierA;

```
mysql> SELECT * FROM SupplierA;
+-----+-----+-----+-----+
| supplierNum | name | status | city |
+-----+-----+-----+-----+
| S1 | Smith | 20 | London |
| S2 | Jones | 10 | Paris |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Q2. Update the Supplier table to reflect a change in Supplier_status to the value '20' for all suppliers from Paris. You should provide a screenshot of the update and of a suitable SELECT statement to show the change. Try searching on Google!

1. Let's view the Supplier table's data before the change:

SELECT * FROM Supplier;

```
mysql> SELECT * FROM Supplier;
+-----+-----+-----+-----+
| supplierNum | name | status | city |
+-----+-----+-----+-----+
| S1 | Smith | 20 | London |
| S2 | Jones | 10 | Paris |
| S3 | Blake | 20 | Paris |
| S4 | Clark | 20 | London |
| S5 | Adams | 30 | Athens |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

2. Perform update:

UPDATE Supplier SET status=20 WHERE city='Paris';

```
mysql> UPDATE Supplier SET status=20 WHERE city='Paris';
Query OK, 1 row affected (0.35 sec)
Rows matched: 2 Changed: 1 Warnings: 0
```

3. Et voila...

`SELECT * FROM Supplier;`

```
mysql> SELECT * FROM Supplier;
+-----+-----+-----+-----+
| supplierNum | name | status | city |
+-----+-----+-----+-----+
| S1 | Smith | 20 | London |
| S2 | Jones | 20 | Paris |
| S3 | Blake | 20 | Paris |
| S4 | Clark | 20 | London |
| S5 | Adams | 30 | Athens |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

Q3. Delete all entries in the Supplies table corresponding to supplierNum S4. You should provide a screenshot of the DELETE statement and of a suitable SELECT statement to show the change.

1. First off, let's view the relevant entries in the Supplies table to pinpoint what exactly we need to delete:

`SELECT * FROM Supplies WHERE supplierNum='S4';`

```
mysql> SELECT * FROM Supplies WHERE supplierNum='S4';
+-----+-----+-----+
| supplierNum | partNum | quantity |
+-----+-----+-----+
| S4 | P2 | 200 |
| S4 | P4 | 300 |
| S4 | P5 | 400 |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

2. Now, delete relevant entries using the DELETE statement:

`DELETE FROM Supplies WHERE supplierNum='S4';`

```
mysql> DELETE FROM Supplies WHERE supplierNum='S4';
Query OK, 3 rows affected (0.00 sec)
```

3. Do these entries (for supplierNum 'S4') still exist?

`SELECT * FROM Supplies WHERE supplierNum='S4';`

```
mysql> SELECT * FROM Supplies WHERE supplierNum='S4';
Empty set (0.00 sec)
```

Q4. Insert the following information into the Supplies table:

supplierNum	partNum	quantity
S4	P2	500
S4	P4	100
S4	P5	800

You should provide a screenshot of the INSERT statement and of a suitable SELECT statement to show the change.

1. Let's try inserting one record of data (the first row) using the following statement-style:

```
INSERT INTO Supplies VALUES ('S4', 'P2', 500);
```

```
mysql> INSERT INTO Supplies VALUES ('S4', 'P2', 500);  
Query OK, 1 row affected (0.05 sec)
```

2. Now, checking if this data is now in its new place:

```
SELECT * FROM Supplies;
```

```
mysql> SELECT * FROM Supplies;  
+-----+-----+-----+  
| supplierNum | partNum | quantity |  
+-----+-----+-----+  
| S1          | P1      | 300      |  
| S1          | P2      | 200      |  
| S1          | P3      | 400      |  
| S1          | P4      | 200      |  
| S1          | P5      | 100      |  
| S1          | P6      | 100      |  
| S2          | P1      | 300      |  
| S2          | P2      | 400      |  
| S3          | P2      | 200      |  
| S4          | P2      | 500      |  
+-----+-----+-----+  
10 rows in set (0.00 sec)
```

3. For learning purposes, we can achieve data entry of the remaining records using another (similar) syntax:

```
INSERT INTO Supplies (supplierNum, partNum, quantity) VALUES  
('S4', 'P4', 100), ('S4', 'P5', 800);
```

```
mysql> INSERT INTO Supplies (supplierNum, partNum, quantity) VALUES
-> ('S4', 'P4', 100), ('S4', 'P5', 800);
Query OK, 2 rows affected (0.00 sec)
Records: 2 Duplicates: 0 Warnings: 0
```

4. Finally, let's view the table, Supplies again to be sure that the data was entered correctly:

```
SELECT * FROM Supplies;
```

```
mysql> SELECT * FROM Supplies;
+-----+-----+-----+
| supplierNum | partNum | quantity |
+-----+-----+-----+
| S1          | P1      | 300      |
| S1          | P2      | 200      |
| S1          | P3      | 400      |
| S1          | P4      | 200      |
| S1          | P5      | 100      |
| S1          | P6      | 100      |
| S2          | P1      | 300      |
| S2          | P2      | 400      |
| S3          | P2      | 200      |
| S4          | P2      | 500      |
| S4          | P4      | 100      |
| S4          | P5      | 800      |
+-----+-----+-----+
12 rows in set (0.00 sec)
```

Q5. Show the count of suppliers in each city ordered from highest to lowest.

1. We can use the following statement toward this end:

```
SELECT city, COUNT(city) AS countSuppliers FROM Supplier GROUP BY city
DESC;
```

```
mysql> SELECT city, COUNT(city) AS countSuppliers FROM Supplier GROUP BY city
DESC;
+-----+-----+
| city    | countSuppliers |
+-----+-----+
| Paris   | 2              |
| London  | 2              |
| Athens  | 1              |
+-----+-----+
3 rows in set (0.00 sec)
```

Q6. Remove the 'weight' attribute from the Parts table.

1. View table prior to making any changes:

```
SELECT * FROM Parts;
```

```
mysql> SELECT * FROM Parts;
+-----+-----+-----+-----+-----+
| partNum | name  | colour | weight | city   |
+-----+-----+-----+-----+-----+
| P1      | Nut   | Red    | 12.0   | London |
| P2      | Bolt  | Green  | 17.0   | Paris  |
| P3      | Screw | Blue   | 17.0   | Oslo   |
| P4      | Screw | Red    | 14.0   | London |
| P5      | Cam   | Blue   | 12.0   | Paris  |
| P6      | Cog   | Red    | 19.0   | London |
+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

2. Perform alteration of table, specifically removing the 'weight' attribute:

```
ALTER TABLE Parts DROP weight;
```

```
mysql> ALTER TABLE Parts DROP weight;
Query OK, 6 rows affected (0.26 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

3. We are now left with the Parts table minus its weight column:

```
SELECT * FROM Parts;
```

```
mysql> SELECT * FROM Parts;
+-----+-----+-----+-----+
| partNum | name  | colour | city   |
+-----+-----+-----+-----+
| P1      | Nut   | Red    | London |
| P2      | Bolt  | Green  | Paris  |
| P3      | Screw | Blue   | Oslo   |
| P4      | Screw | Red    | London |
| P5      | Cam   | Blue   | Paris  |
| P6      | Cog   | Red    | London |
+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

Q7. Show only the name of all the parts in the Parts table except red parts.

1. Select all information from the Parts table to begin with:

`SELECT * FROM Parts;`

```
mysql> SELECT * FROM Parts;
+-----+-----+-----+-----+
| partNum | name  | colour | city  |
+-----+-----+-----+-----+
| P1      | Nut   | Red    | London |
| P2      | Bolt  | Green  | Paris  |
| P3      | Screw | Blue   | Oslo   |
| P4      | Screw | Red    | London |
| P5      | Cam   | Blue   | Paris  |
| P6      | Cog   | Red    | London |
+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

2. Now, select only the name of all parts where the name is not 'red':

`SELECT name FROM Parts WHERE color <> 'red';`

```
mysql> SELECT name FROM Parts WHERE colour <> 'red';
+-----+
| name  |
+-----+
| Bolt  |
| Screw |
| Cam   |
+-----+
3 rows in set (0.00 sec)
```

Q8. Show all the parts in the Parts table except those with a quantity of 500 or 800 using a NOT IN condition.

1. Use following command to perform this operation:

`SELECT Parts.partnum, name, colour, quantity, city FROM Parts, Supplies WHERE Parts.partNum = Supplies.partNum AND quantity NOT IN (500, 800);`

```
mysql> SELECT Parts.partnum, name, colour, quantity, city FROM Parts, Supplies W
HERE Parts.partNum = Supplies.partNum AND quantity NOT IN (500, 800);
```

partnum	name	colour	quantity	city
P1	Nut	Red	300	London
P2	Bolt	Green	200	Paris
P3	Screw	Blue	400	Oslo
P4	Screw	Red	200	London
P5	Cam	Blue	100	Paris
P6	Cog	Red	100	London
P1	Nut	Red	300	London
P2	Bolt	Green	400	Paris
P2	Bolt	Green	200	Paris
P4	Screw	Red	100	London

10 rows in set (0.32 sec)

Q9. Show all entries from the Supplies table with their corresponding part names.

1. Providing all existing information from Supplies table, plus parts' corresponding names;

```
SELECT supplierNum, Supplies.partNum, name, quantity,
FROM Supplies LEFT OUTER JOIN Parts ON Supplies.partNum=Parts.partNum;
```

```
mysql> SELECT supplierNum, Supplies.partNum, name, quantity FROM Supplies LEFT
-> OUTER JOIN Parts ON Supplies.partNum=Parts.partNum;
```

supplierNum	partNum	name	quantity
S1	P1	Nut	300
S1	P2	Bolt	200
S1	P3	Screw	400
S1	P4	Screw	200
S1	P5	Cam	100
S1	P6	Cog	100
S2	P1	Nut	300
S2	P2	Bolt	400
S3	P2	Bolt	200
S4	P2	Bolt	500
S4	P4	Screw	100
S4	P5	Cam	800

12 rows in set (0.00 sec)

Q10. Show the names of all Suppliers that appear more than once in the Supplies table.

1. Method 1:

```
SELECT name, count(Supplies.supplierNum) AS countSuppliers FROM Supplier LEFT  
JOIN Supplies ON Supplier.supplierNum=Supplies.supplierNum GROUP BY name  
HAVING (countSuppliers) >1;
```

```
mysql> SELECT name, count(Supplies.supplierNum) AS countSuppliers FROM Supplier  
LEFT JOIN Supplies ON Supplier.supplierNum=Supplies.supplierNum GROUP BY name HA  
VING (countSuppliers) > 1;  
+-----+-----+  
| name | countSuppliers |  
+-----+-----+  
| Clark | 3 |  
| Jones | 2 |  
| Smith | 6 |  
+-----+-----+  
3 rows in set (0.00 sec)
```

2. Method 2 where only names are included:

```
SELECT name FROM Supplier INNER JOIN Supplies ON  
Supplier.supplierNum=Supplies.supplierNum GROUP BY name  
HAVING COUNT(name) > 1;
```

```
mysql> SELECT name FROM Supplier INNER JOIN Supplies ON Supplier.supplierNum=Sup  
plies.supplierNum GROUP BY name HAVING COUNT(name) > 1;  
+-----+  
| name |  
+-----+  
| Clark |  
| Jones |  
| Smith |  
+-----+  
3 rows in set (0.00 sec)
```