

CCINFOM

Module 2 Lesson 01: Experiencing Databases using MYSQL and MYSQL Workbench

Estimated Time to consume material – 1.5 Hours

To better understand what databases are, for this week, we will be focusing on experiencing databases through the use of MYSQL Workbench to access and interact with a database created in MYSQL. At the end of the week, hopefully through this experience and a formal lecture, we will have a better understanding of the different software, hardware and human elements of the database environment.

Requirements for this lesson

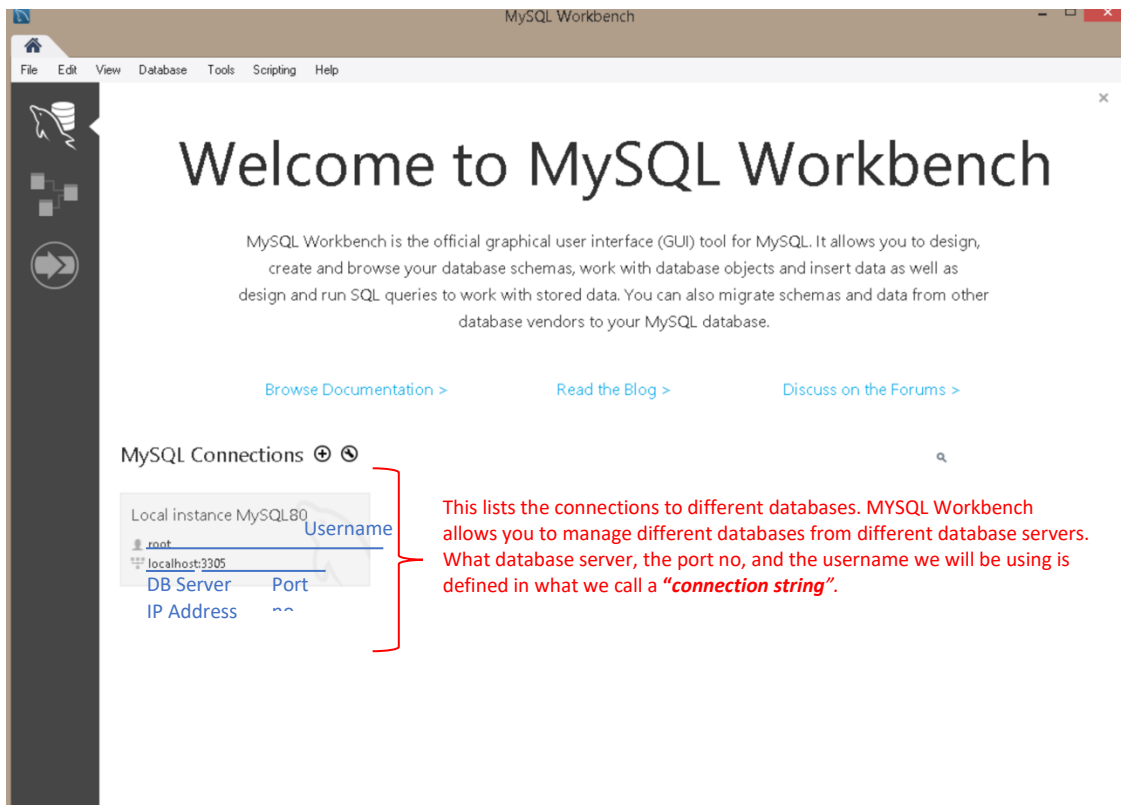
1. Installed MYSQL and MYSQL Workbench
2. Downloaded MYSQL Script File (ccinfomdemo.sql and dbworld.sql)

Tips for Learning

1. While going through this course material, it is best that you write down your questions, especially on the ideas that are introduced in this material
2. When something went wrong, read the error messages completely, try to check what you have done, correct it if necessary, if something is still wrong, take note of it and raise it during our synchronous session

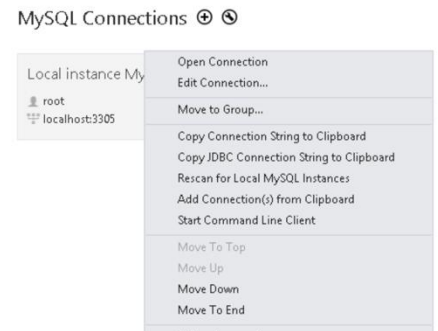
Part 1: Running and Familiarizing yourself with MYSQL WORKBENCH

1. Run MYSQL Workbench. The first screen that will appear is shown below. From the main screen, we can see MySQL Connections.

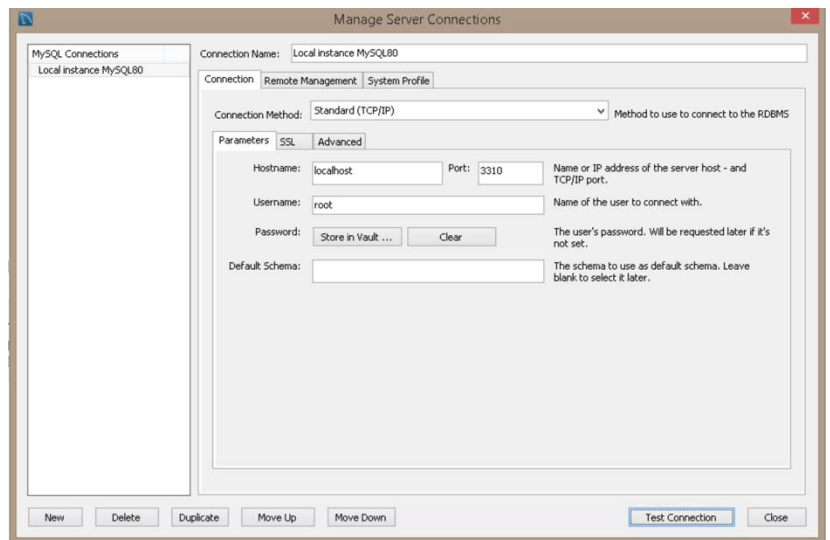


2. Take note that the port no we configured your workstations at, is 3310. Localhost simply means that the database server we will be connecting to the database server installed in your workstation. If the database is not within our workstations, we have to specify the IP Address of the database server.
3. Root is a superuser account. This is usually reserved to be used by the Database Administrator to manage databases in a database server. We are only using root in the course to remove restrictions in our use of databases. Access to data is power, and definitely we do not want unlimited access to data to be in the hands of many users. In practice, database administrators will be creating an account for developers and the application to be developed, to restrict access to databases.

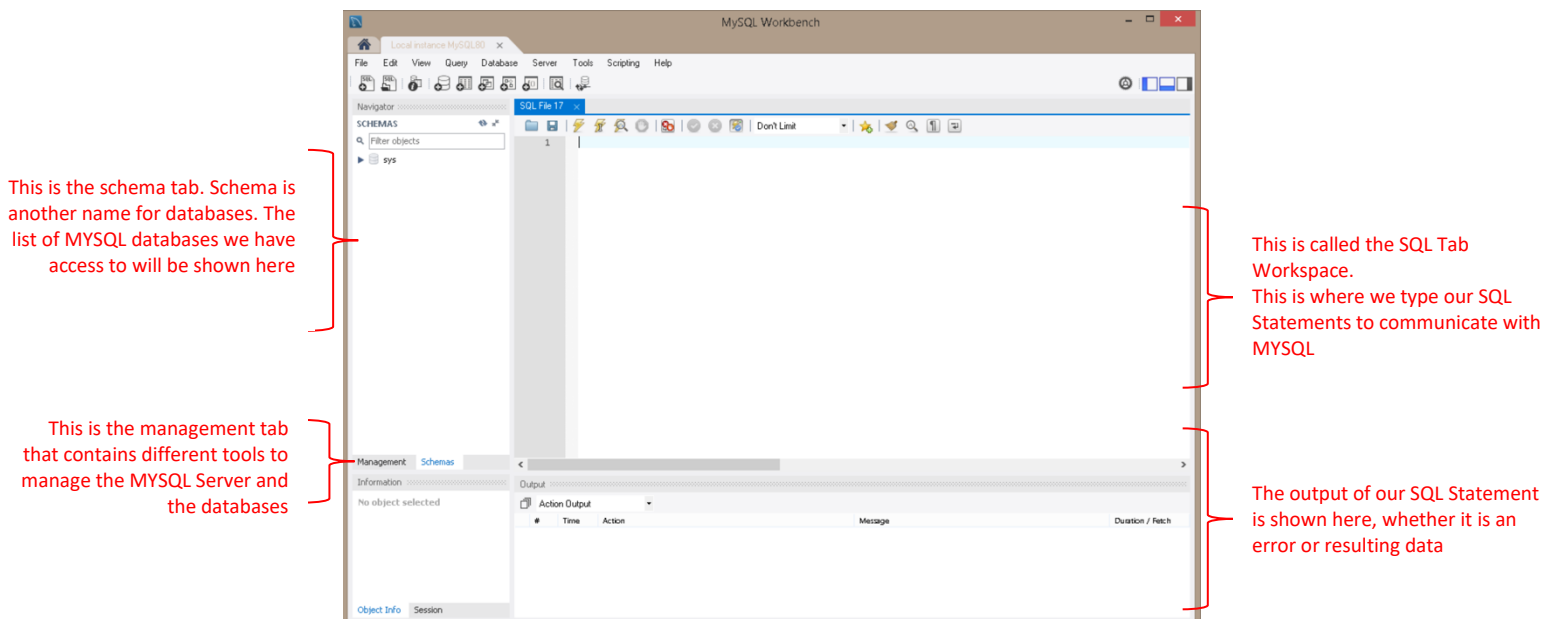
4. If you need to edit the connection to change its configuration (to conform to our configuration), you can right-click the connection and choose **“Edit Connection”** from the menu.



5. Change the hostname, port and username to localhost, 3306 and root respectively. Click on **“Test Connection”** to validate if our configuration is correct.
6. If the connection test failed, then the following may have happened
 - a. You may not have MYSQL, or your MYSQL has not been properly started. Get in touch with your teacher as soon as possible in order to assess your DB and check for a solution
 - b. You might have mistakenly entered a wrong configuration, check it again



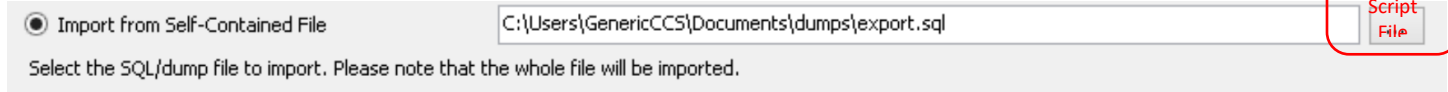
7. If the connection test is successful, then you are ready to access MYSQL. To access MYSQL in your workstation, just click your connection. The screen below will appear. Familiarize yourself with the different parts of the screen. On your workstation, there may appear already some databases because when we install MYSQL on your workstation, we included Samples and Examples.



Part 2: Creating a new Database from a Database Script

1. Now that you are familiar with some of the parts of MYSQL Workbench, we can now create a new database using a database script. A **database script** is a text file containing a set of SQL Statements that will create structures and initial contents of the database. Since database is a storage technology, it is composed of (at the minimum) storage structures and content (or data). Advance database scripts may contain programming codes to implement advance structures in the target database. MYSQL manages **relational databases**, therefore its primary storage structure is called a table. To create a new database using a database script, follow the following simple steps:
 - a. Click on the management tab
 - b. Choose from the available tools, “Data Import/Restore”
 - c. Under import options, choose “Import from self-contained file”

Click
here to
select
our
Script
File



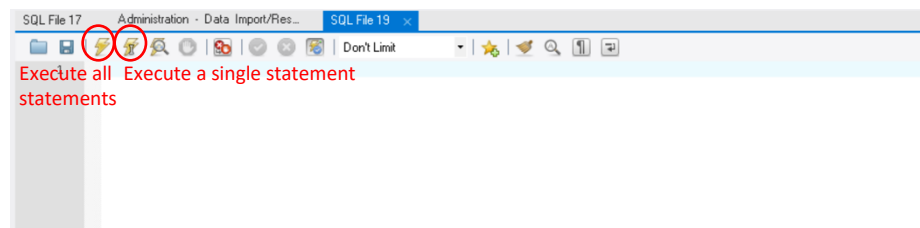
- d. Click on the select file button and select the script file **ccinfomdemo.sql** from the directory you saved it to. (Clicking on open will select the file)
- e. Once the file is selected, click on the “Start Import” button
- f. Wait until the message that the importing of the file has finished. Once finished, click on the “Schemas” tab.
- g. Press the schema refresh button, to refresh the schema list.
- h. You should be seeing our newly created database named **ccinfomdemo**



2. At any given time, we can only work on one (1) database. To work on a database, we have to make it the active database. To do this, just double-click on the database name and its font will become **BOLD**, this means it is the active database. A lot of students' experience in the past whereby SQL statements are not executing correctly is caused by not activating the correct database. It is best that from time to time check if the database you want to work on is the active database.
3. At this point, we have our database created, let's explore what is inside our database. It is difficult to work on a database that we do not know what it contains. Expand the database **ccinfomdemo** and you will see the four (4) fundamental structures inside a MYSQL database – Table, Views, Stored Procedures and Functions. Stored Procedures and Functions are programs inside databases. These are coded by Database Developers to add behavior on databases. An example could be a set of actions that will be performed by the database at the end of the month. This set of actions may include a series of updating records and computations that keeps on being repeated so it was made as a function. BSIT students take an advance database course that handles more of the tasks performed by a database developer. BSCS students take an advance database course to investigate more underlying functionality of databases.
4. Let's expand tables, the only structure created in our database. You will see three tables – employee, itemrequest and items. Tables are structures where data are stored. Specific kind of data being stored on the table is defined by its columns. If you expand a table, for example – employee, you will see that a table has columns, index, foreign keys and triggers. And expanding columns will reveal – username, password, completename, currentpoints. Columns simply means that every record in the employee table will contain data about username, password, completename and current points.
5. For our lesson, we need to understand up to that point for now. Databases have Tables, and Tables have columns.

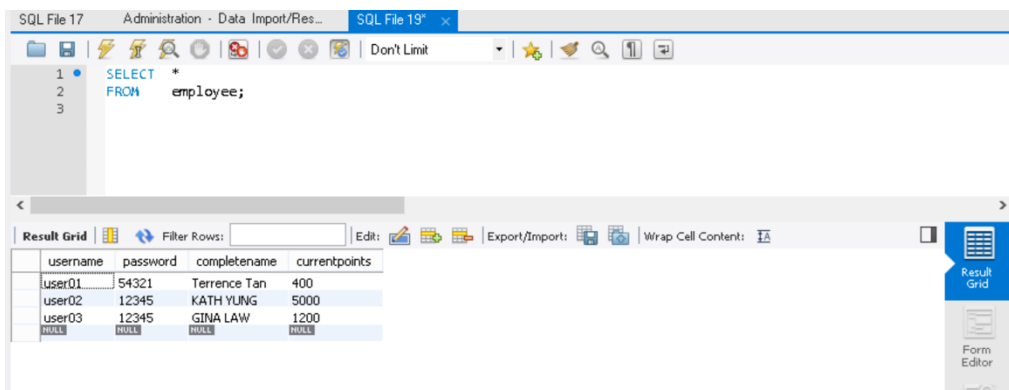
Part 3: Basic Interactions with Databases

1. How do we interact with Databases? And what exactly do we mean by interacting with databases? Interacting with database means communicating with the database what we want to happen – may it be getting data, saving the data, modifying data, creating a new table, changing the columns of a table. Simply, interacting with the database is giving instructions to it. And any act of giving instructions to a computing technology requires a language; and for relational databases, it will be – Structured Query Language (SQL).
2. For this lesson, we will focus on one kind of SQL, the Data Manipulation Language (SQL-DML). SQL-DML is a set of instructions that allows us to perform (a) Get Data from tables inside our database and (b) Modify, Create, Delete records in tables inside our database. To send an instruction to databases, we simply write an SQL statement and execute it. And for this lesson, we will only focus on getting data. In future lessons, we will be covering other instructions as well as the other kind of SQL.
3. SQL Statements can be written and executed using the SQL Query Tab. To create a new SQL Query Tab, press Ctrl-T. Before we start writing and executing SQL Statements, lets understand first what is available to us in a Query tab. Since you can have several Query Tabs, the active query tab is highlighted in blue. You can associate a query tab to a paper where you can write statements. Sometimes we want to group our statements together, so we put them on the same paper, and in MYSQL Workbench, same Query Tab. There are two ways to execute statements – execute all statements or execute a single statement. Executing a single statement is done by putting the cursor at the start of a statement and pressing the single execute statement button.



4. Let's write an SQL Statement that will get all the records of employees. Type the SQL statement below in your Query Tab and press the execute single statement button. The result of the SQL statement execution is shown below. The row full of null simply means the end of results. SELECT * means get all the columns of records of the table indicated in the FROM clause.

```
SELECT      *
FROM        employee;
```



5. Try doing the same SQL Statement, but this time getting all the records of items. Type the SQL statement below in your Query Tab and press the execute single statement button. Do not forget to put your cursor at the start or any part of the SQL statement you are executing.

```
SELECT      *
FROM        items
```

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the statement: `SELECT * FROM items;`. The Result Grid below shows the following data:

itemkind	itemno	itemname	points	qty
Gadoet	GAD001	Voice Recorder	2000	49
Gadoet	GAD002	Bluetooth Mouse	1000	43
Gadoet	GAD003	Basic Phone	300	76
Gadoet	GAD004	Label Maker	900	83
Kitchen	KIT001	Black Plate Set	1000	95
Kitchen	KIT002	Black Utensils Set	950	38
Kitchen	KIT003	Elegant Knife Set	890	21
Kitchen	KIT004	Juicer	3000	5
Office Supply	OFC001	Notebook A5 3-Set	250	720
Office Supply	OFC002	Marker 3-Color Set	200	63
Office Supply	OFC003	Folder Multicolor 25ocs Set	100	182
Office Supply	OFC004	National Bookstore 100 GC	130	82

6. SQL is a powerful language. You can do more than just getting all the data. Let's try another SQL statement. Type the SQL statement below in your Query Tab and press the execute single statement button. What did you notice in the result? It fetches records of items where the points are greater than 500.

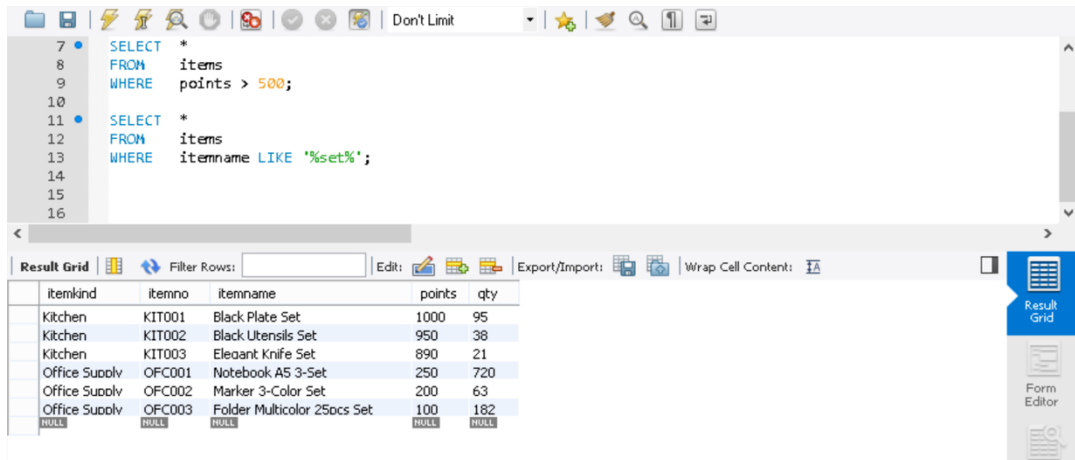
```
SELECT      *
FROM        items
WHERE       points > 500;
```

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the statement: `SELECT * FROM items WHERE points > 500;`. The Result Grid below shows the filtered data:

itemkind	itemno	itemname	points	qty
Gadoet	GAD001	Voice Recorder	2000	49
Gadoet	GAD002	Bluetooth Mouse	1000	43
Gadoet	GAD004	Label Maker	900	83
Kitchen	KIT001	Black Plate Set	1000	95
Kitchen	KIT002	Black Utensils Set	950	38
Kitchen	KIT003	Elegant Knife Set	890	21
Kitchen	KIT004	Juicer	3000	5
NULL	NULL	NULL	NULL	NULL

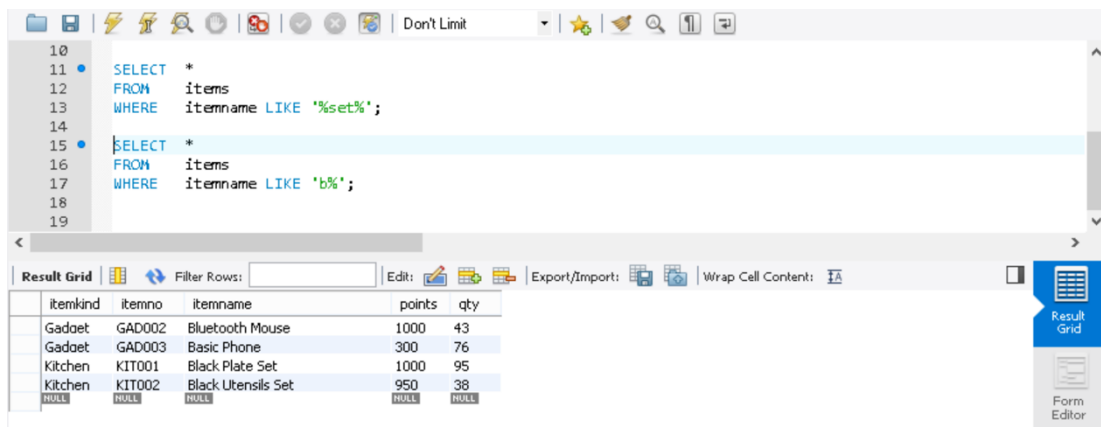
7. Let's try another SQL statement. Type the SQL statement below in your Query Tab and press the execute single statement button. What did you notice in the result? It fetched records of items where the itemname contains the word "set".

```
SELECT      *
FROM        items
WHERE       itemname LIKE '%set%';
```



8. Let's try another SQL statement. Type the SQL statement below in your Query Tab and press the execute single statement button. What did you notice in the result? It fetched records of items where the itemname starts with "b". If you want to get records of items where the itemname ends with "t", the where clause could just be changed to `WHERE itemname LIKE '%t'`. Why don't you try it and see what the results will look like.

```
SELECT      *
FROM        items
WHERE       itemname LIKE 'b%';
```



If you tried it, you will have a result as shown below:

itemkind	itemno	itemname	points	qty
Kitchen	KIT001	Black Plate Set	1000	95
Kitchen	KIT002	Black Utensils Set	950	38
Kitchen	KIT003	Elegant Knife Set	890	21
Office Supply	OFC001	Notebook A5 3-Set	250	720
Office Supply	OFC002	Marker 3-Color Set	200	63
Office Supply	OFC003	Folder Multicolor 25ocs Set	100	182
NULL	NULL	NULL	NULL	NULL

9. Let's try another SQL statement. Type the SQL statement below in your Query Tab and press the execute single statement button. What did you notice in the result? It fetched records of items where the points is from 200 to 890. BETWEEN is inclusive of the upper value and the lower value. If you want something that will get the records of items that are above 200 and below 890, then you have to change the where clause to *WHERE points > 200 AND points < 890*. Why don't you try it and see what the results will look like.

```
SELECT      *
FROM        items
WHERE       points between 200 and 890;
```

The screenshot shows the MySQL Workbench interface. The SQL editor contains the query: `SELECT * FROM items WHERE points BETWEEN 200 and 890;`. The Result Grid below shows the following data:

itemkind	itemno	itemname	points	qty
Gadaget	GAD003	Basic Phone	300	76
Kitchen	KIT003	Elegant Knife Set	890	21
Office Supplv	OFC001	Notebook A5 3-Set	250	720
Office Supplv	OFC002	Marker 3-Color Set	200	63

If you tried it, you will have a result as shown below. Notice no results with 200 and 890 points.

itemkind	itemno	itemname	points	qty
Gadaget	GAD003	Basic Phone	300	76
Office Supplv	OFC001	Notebook A5 3-Set	250	720
			NULL	NULL

10. What if we want to get the total points per items that can still be earned? Let's try another SQL statement. Type the SQL statement below in your Query Tab and press the execute single statement button. What do you observe from the result? It added a new column containing a mathematical computation of $points * qty$. That is correct, you can have mathematical computations to be performed per records in SQL. Since it's not good to have a column name to be a mathematical equation, you can rename it by using the AS clause. For example, you can instead type *points*qty AS totalpoints*. Try it.

```
SELECT      *, points*qty
FROM        items;
```

The screenshot shows the MySQL Workbench interface. The SQL editor contains the query: `SELECT *, points*qty FROM items;`. The Result Grid below shows the following data:

itemkind	itemno	itemname	points	qty	points*qty
Gadaget	GAD001	Voice Recorder	2000	49	98000
Gadaget	GAD002	Bluetooth Mouse	1000	43	43000
Gadaget	GAD003	Basic Phone	300	76	22800
Gadaget	GAD004	Label Maker	900	83	74700
Kitchen	KIT001	Black Plate Set	1000	95	95000
Kitchen	KIT002	Black Utensils Set	950	38	36100
Kitchen	KIT003	Elegant Knife Set	890	21	18690
Kitchen	KIT004	Juicer	3000	5	15000
Office Supplv	OFC001	Notebook A5 3-Set	250	720	180000
Office Supplv	OFC002	Marker 3-Color Set	200	63	12600
Office Supplv	OFC003	Folder Multicolor 25ocs Set	100	182	18200
Office Supplv	OFC004	National Bookstore 100 GC	130	82	10660

If you have tried changing the statement, you will be getting the result as shown below.

```
SELECT      *, points*qty AS totalpoints
FROM        items;
```

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```

25 WHERE points BETWEEN 200 and 890;
26
27 SELECT *
28 FROM items
29 WHERE points > 200 and points < 890;
30
31 SELECT *, points*qty
32 FROM items;
33
34 SELECT *, points*qty as totalpoints
35 FROM items;
36

```

The Result Grid shows the following data:

itemkind	itemno	itemname	points	qty	totalpoints
Gadget	GAD001	Voice Recorder	2000	49	98000
Gadget	GAD002	Bluetooth Mouse	1000	43	43000
Gadget	GAD003	Basic Phone	300	76	22800
Gadget	GAD004	Label Maker	900	83	74700
Kitchen	KIT001	Black Plate Set	1000	95	95000
Kitchen	KIT002	Black Utensils Set	950	38	36100
Kitchen	KIT003	Elegant Knife Set	890	21	18690
Kitchen	KIT004	Juicer	3000	5	15000
Office Supplv	OFC001	Notebook A5 3-Set	250	720	180000
Office Supplv	OFC002	Marker 3-Color Set	200	63	12600
Office Supplv	OFC003	Folder Multicolor 25ocs Set	100	182	18200
Office Supplv	OFC004	National Bookstore 100 GC	130	82	10660

11. Before we try other SQL Statements. Write and execute an SQL Statement that will get all the records of item requests.

```
SELECT      *
FROM        itemrequest;
```

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```

19 SELECT *
20 FROM items
21 WHERE itemname LIKE '%t';
22
23 SELECT *
24 FROM itemrequest;
25
26
27
28

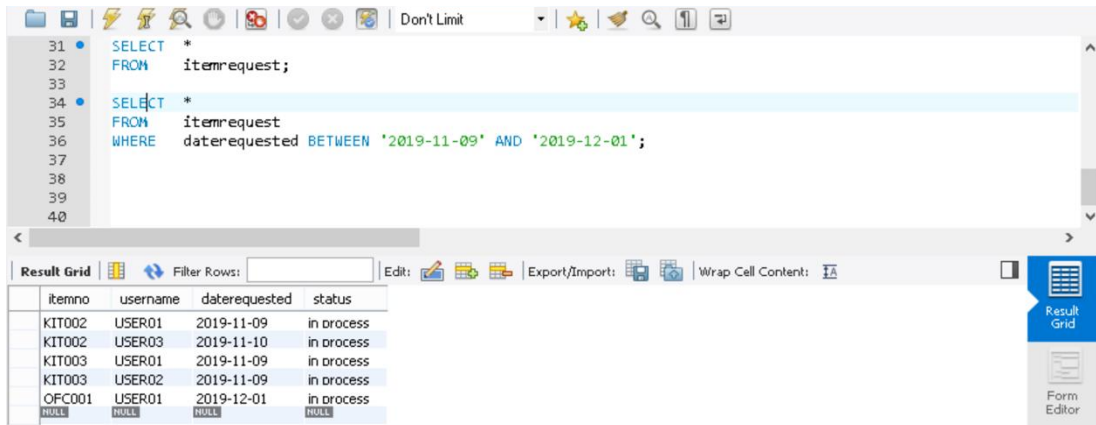
```

The Result Grid shows the following data:

itemno	username	daterequested	status
GAD001	USER01	2019-11-07	in process
GAD001	USER02	2019-11-08	in process
GAD002	USER03	2019-11-08	in process
GAD004	USER01	2019-11-08	in process
KIT002	USER01	2019-11-09	in process
KIT002	USER03	2019-11-10	in process
KIT003	USER01	2019-11-09	in process
KIT003	USER02	2019-11-09	in process
OFC001	USER01	2019-12-01	in process
OFC002	USER01	2019-12-02	in process
OFC003	USER01	2019-12-02	in process
NULL	NULL	NULL	NULL

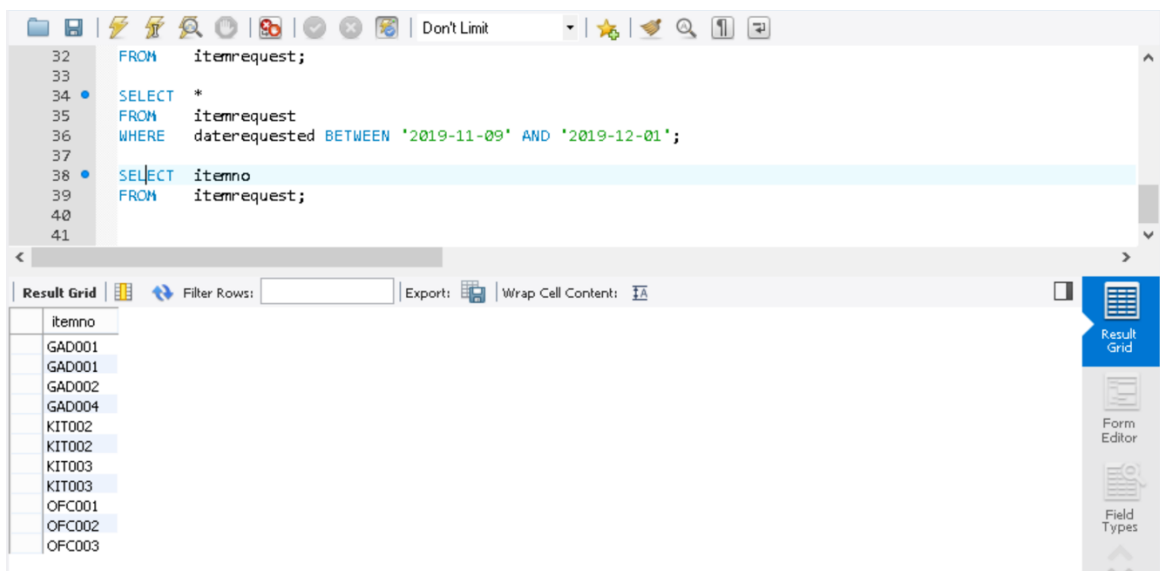
12. Let's try another SQL statement. Type the SQL statement below in your Query Tab and press the execute single statement button. What did you notice in the result? It fetched records of itemrequest between 2019-11-09 and 2019-12-01. This shows that BETWEEN also works on dates, and not just on numbers.

```
SELECT      *
FROM        itemrequest
WHERE       daterequested BETWEEN '2019-11-09' AND '2019-12-01';
```



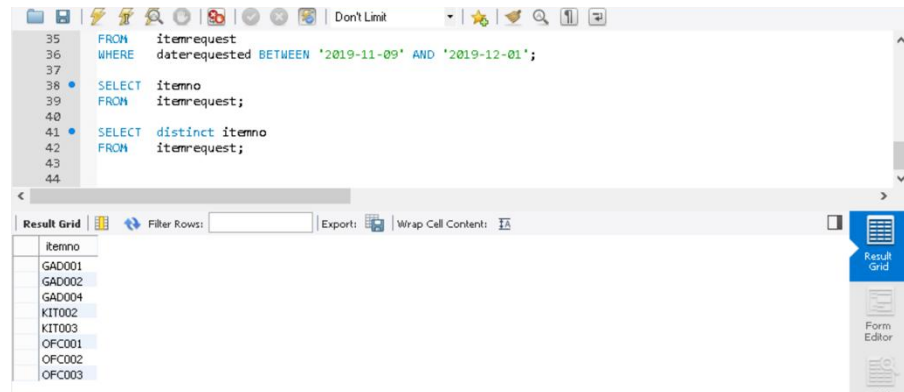
13. Let's try another SQL statement. Type the SQL statement below in your Query Tab and press the execute single statement button. What did you notice in the result? It fetched the itemno of the records of itemrequest. In context of the records, it returns the list of items that were requested. In SQL, you don't need to always retrieve all the columns, you can specify the column/s you want to get.

```
SELECT      itemno
FROM        itemrequest;
```



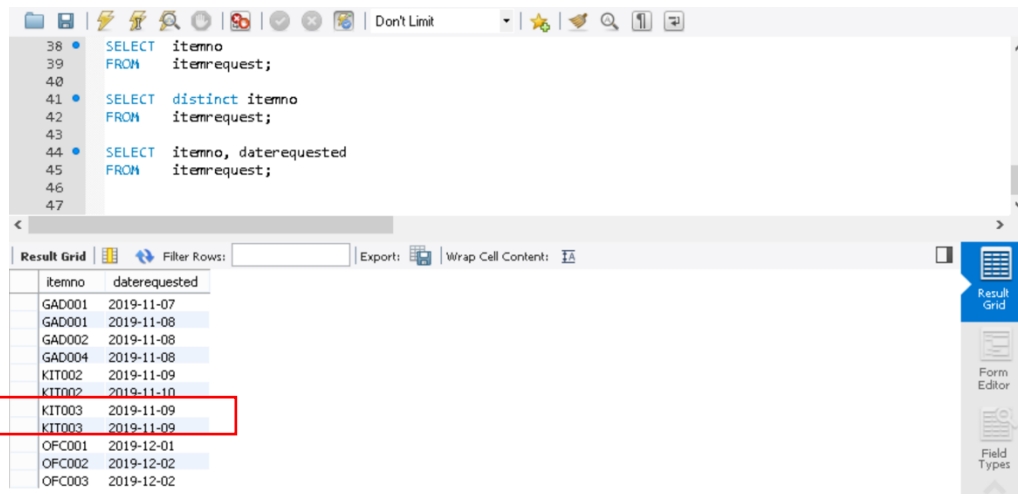
14. What do you notice from the results of the previous SQL Statement? It did list the items that were requested. Since some items are requested by several users, you will notice that the item repeats in the list. That doesn't appear to be a presentable output, does it? Type the SQL statement below in your Query Tab and press the execute single statement button. What did you notice in the result? It is still a list of items requested but this time, there are no duplicates.

```
SELECT      DISTINCT itemno
FROM        itemrequest;
```



15. Let's try another SQL statement. Type the SQL statement below in your Query Tab and press the execute single statement button. What did you notice in the result? It returns the itemno and daterequested of records of itemrequest. And since the same item can be requested by different users on the same date, it is possible that our result will contain the same itemno and daterequested appearing more than once (as highlighted in red rectangle below). To address this problem, we use DISTINCT. We just replace the select clause with SELECT DISTINCT itemno, daterequested. Try it and see what happens.

```
SELECT      itemno, username
FROM        itemrequest;
```



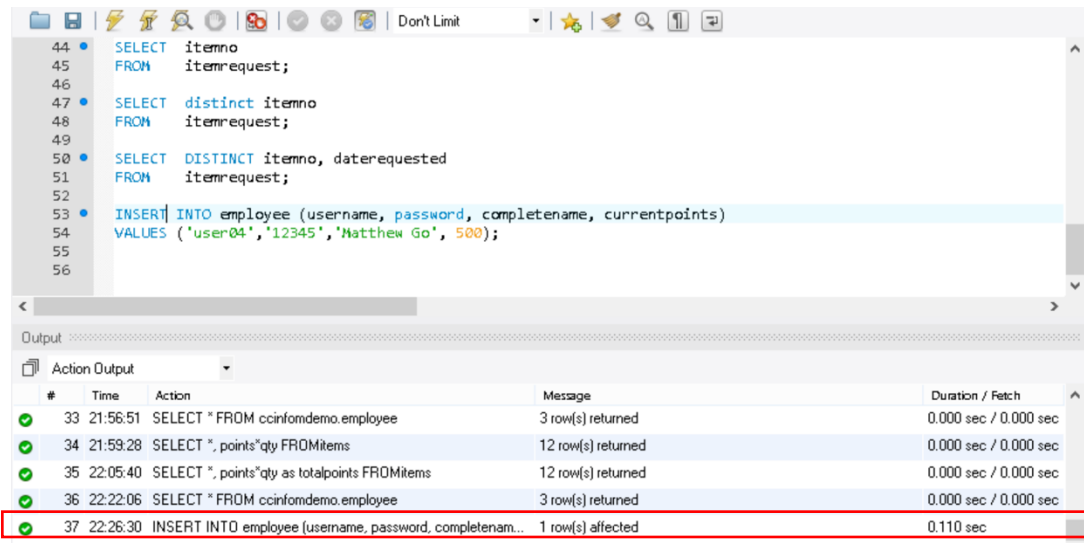
If you tried it, you will have a result as shown below. Notice no more duplicates.

itemno	daterequested
GAD001	2019-11-07
GAD001	2019-11-08
GAD002	2019-11-08
GAD004	2019-11-08
KIT002	2019-11-09
KIT002	2019-11-10
KIT003	2019-11-09
OFC001	2019-12-01
OFC002	2019-12-02
OFC003	2019-12-02

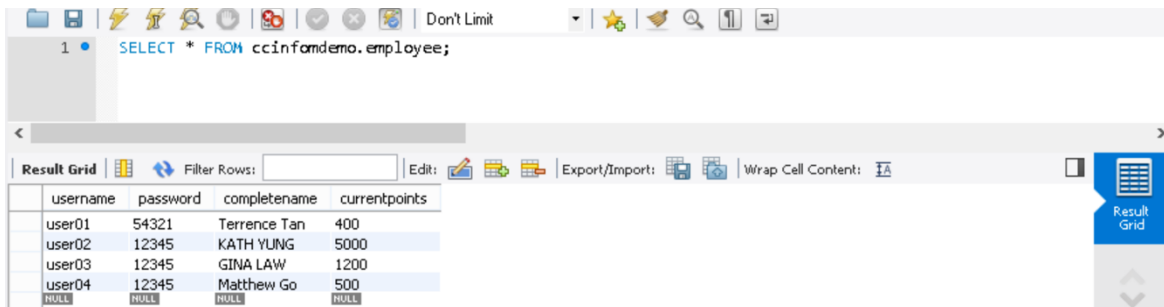
Part 4: Creating, Modifying and Deleting Records

1. As we mentioned in succeeding lesson, SQL-DML, aside from getting data, can also be used to create, modify and delete records from a table. SQL-DML has the INSERT, UPDATE and DELETE statements to create, modify, and delete records respectively.
2. To create a new record for example in employee, type the SQL statement below. The output message as shown below shows that 1 row of record was added.

```
INSERT INTO employee (username, password, completename, currentpoints)
VALUES ('user04','12345','Matthew Go', 500);
```

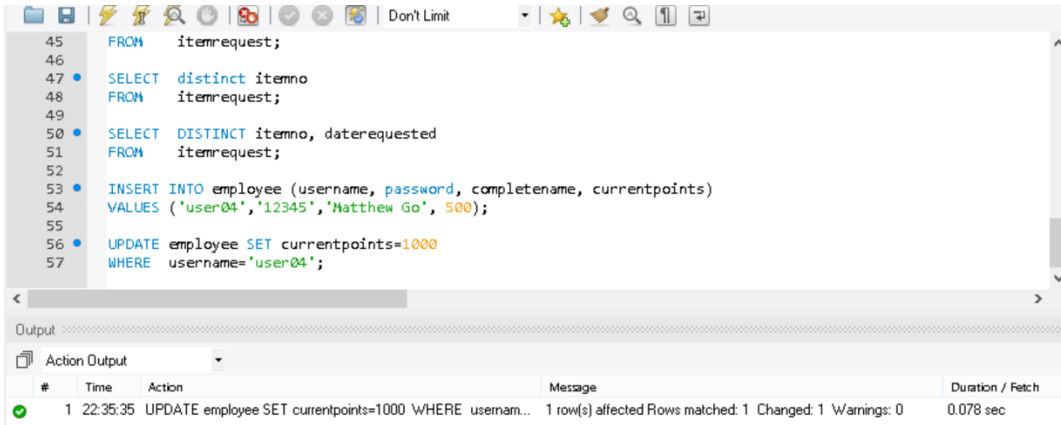


To check if the new record was indeed added to the employee table, retrieve all the records of employee. Did you see our new record?

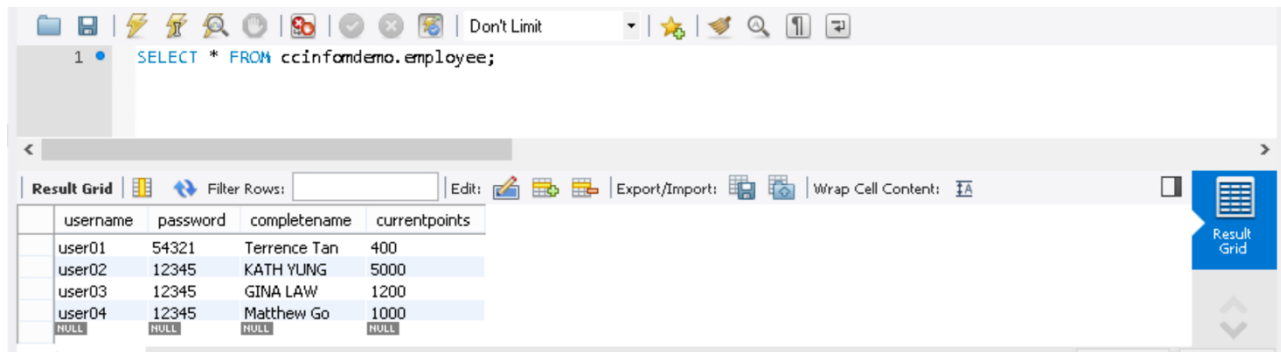


3. To modify an existing record, for example the current points of Matthew Go to be changed to 1000, type the SQL Statement below. The output message as shown below shows that 1 row of record was modified.

```
UPDATE employee SET currentpoints=1000
WHERE username='user04';
```

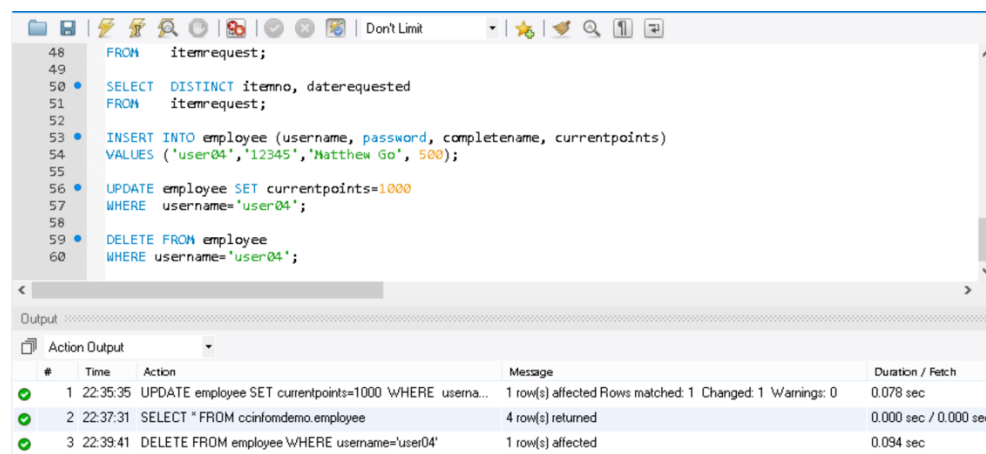


To check if the new record was indeed added to the employee table, retrieve all the records of employee. Did you see our record modified?



4. To delete an existing record, for example delete the record of Matthew, type the SQL Statement below. The output message as shown below shows that 1 row of record was deleted.

```
DELETE FROM employee
WHERE username='user04';
```





To check if the new record was indeed added to the employee table, retrieve all the records of employee. Did you see our record deleted?








EXERCISE

Let us have an exercise on what we have learned from the lessons above. First, create a new database using the script file – dbworld.sql. Using the dbworld DB, write the SQL Statement that will satisfy each data requirement below. Collect your SQL Statements in a notepad file and name the file using your lastname-firstname.sql. Just be ready with the file, in case your teacher will collect it for formative assessment.








1. Get all the cities of Thailand
2. Get all the cities of Thailand with a population of more than 100,000
3. Get all the countries with a life expectancy of 70 to 80 years old
4. Get all the continents with countries having a population below 1,000
5. Get all the countries (name, continent, governmentform) that are Republics
6. Get all the countries that has celebrated at least 100 years of independence
7. Get all the districts of Bangladesh
8. Get all the countries where more than 80% speaks Arabic

EXERCISE (OUTPUTS)








1. Get all the cities of Thailand (Returns 12 rows)
- 2.

Result Grid   Filter Rows: <input type="text" value="Search"/> Edit:    Export/Import:  					
ID	Name	CountryCode	District	Population	
3320	Bangkok	THA	Bangkok	6320174	
3321	Nonthaburi	THA	Nonthaburi	292100	
3322	Nakhon Ratchasima	THA	Nakhon Ratchasima	181400	
3323	Chiang Mai	THA	Chiang Mai	171100	
3324	Udon Thani	THA	Udon Thani	158100	
3325	Hat Yai	THA	Songkhla	148632	
3326	Khon Kaen	THA	Khon Kaen	126500	
3327	Pak Kret	THA	Nonthaburi	126055	
3328	Nakhon Sawan	THA	Nakhon Sawan	123800	
3329	Ubon Ratchathani	THA	Ubon Ratchathani	116300	
3330	Songkhla	THA	Songkhla	94900	
3331	Nakhon Pathom	THA	Nakhon Pathom	94100	




3. Get all the cities of Thailand with a population of more than 100,000 (Returns 10 rows)

Result Grid   Filter Rows: <input type="text" value="Search"/> Edit:    Export/Import:  					
ID	Name	CountryCode	District	Population	
3320	Bangkok	THA	Bangkok	6320174	
3321	Nonthaburi	THA	Nonthaburi	292100	
3322	Nakhon Ratchasima	THA	Nakhon Ratchasima	181400	
3323	Chiang Mai	THA	Chiang Mai	171100	
3324	Udon Thani	THA	Udon Thani	158100	
3325	Hat Yai	THA	Songkhla	148632	
3326	Khon Kaen	THA	Khon Kaen	126500	
3327	Pak Kret	THA	Nonthaburi	126055	
3328	Nakhon Sawan	THA	Nakhon Sawan	123800	
3329	Ubon Ratchathani	THA	Ubon Ratchathani	116300	




4. Get all the countries with a life expectancy of 70 to 80 years old (Returns 107 rows)

Result Grid   Filter Rows: <input type="text" value="Search"/> Edit:    Export/Import:  										
Code	Name	Continent	Region	SurfaceArea	IndepYear	Population	LifeExpectan...	GNP	GNPOld	LocalName
ABW	Aruba	North America	Caribbean	193.00	NULL	103000	78.4	828.00	793.00	Aruba
ALA	Anguilla	North America	Caribbean	96.00	NULL	8000	76.1	63.20	NULL	Anguilla
ALB	Albania	Europe	Southern Europe	28748.00	1912	3401200	71.6	3205.00	2500.00	Shqipëria
ANT	Netherlands Antilles	North America	Caribbean	800.00	NULL	217000	74.7	1941.00	NULL	Nederlandse Antillen
ARE	United Arab Emirates	Asia	Middle East	83600.00	1971	2441000	74.1	37966.00	36846.00	Al-Imarat al-Arabiya al-Muttahida
ARG	Argentina	South America	South America	2780400.00	1816	37032000	75.1	340238.00	323310.00	Argentina
ASM	American Samoa	Oceania	Polynesia	199.00	NULL	68000	75.1	334.00	NULL	Amerika Samoa
ATG	Antigua and Barbuda	North America	Caribbean	442.00	1981	68000	70.5	612.00	584.00	Antigua and Barbuda
AUS	Australia	Oceania	Australia and New Zealand	7741220.00	1901	18886000	79.8	351182.00	392911.00	Australia
AUT	Austria	Europe	Western Europe	83859.00	1918	8091800	77.7	211860.00	206025.00	Ästerreich
BEL	Belgium	Europe	Western Europe	30518.00	1830	10239000	77.8	249704.00	243948.00	België/Belgique
BGR	Bulgaria	Europe	Eastern Europe	110994.00	1908	8190900	70.9	12178.00	10169.00	Balgarija




5. Get all the continents with countries having a population below 1,000 (Returns 3 rows)

Result Grid   Filter Rows: <input type="text" value="Search"/> Export: 	
continent	
Antarctica	
Oceania	
Africa	




6. Get all the countries (name, continent, governmentform) that are Republics (Returns 143 rows)

Result Grid   Filter Rows: <input type="text" value="Search"/> Export: 			
	name	continent	governmentform
▶	Angola	Africa	Republic
	Albania	Europe	Republic
	Argentina	South America	Federal Republic
	Armenia	Asia	Republic
	Austria	Europe	Federal Republic
	Azerbaijan	Asia	Federal Republic
	Burundi	Africa	Republic
	Benin	Africa	Republic
	Burkina Faso	Africa	Republic
	Bangladesh	Asia	Republic
	Bulgaria	Europe	Republic




7. Get all the countries that has celebrated at least 100 years of independence (Returns 62 rows)

Result Grid   Filter Rows: <input type="text" value="Search"/> Export: 			
	name	indepyear	YearsIndependence...
▶	Afghanistan	1919	102
	Albania	1912	109
	Andorra	1278	743
	Argentina	1816	205
	Australia	1901	120
	Austria	1918	103
	Belgium	1830	191
	Bulgaria	1908	113
	Bolivia	1825	196
	Brazil	1822	199
	Bhutan	1910	111

8. Get all the districts of Bangladesh (Returns 6 rows)

Result Grid   Filter Rows: <input type="text" value="Search"/> Export: 	
	district
▶	Dhaka
	Chittagong
	Khulna
	Rajshahi
	Barisal
	Sylhet

9. Get all the countries where more than 80% speaks Arabic (Returns 10 rows)

Result Grid   Filter Rows: <input type="text" value="Search"/> Export: 			
	name	language	percentage
▶	Algeria	Arabic	86.0
	Egypt	Arabic	98.8
	Western Sahara	Arabic	100.0
	Jordan	Arabic	97.9
	Lebanon	Arabic	93.0
	Libyan Arab Jamahiriya	Arabic	96.0
	Palestine	Arabic	95.9
	Saudi Arabia	Arabic	95.0
	Syria	Arabic	90.0
	Yemen	Arabic	99.6