PART A

Question 1

To create Table 1 I used the following commands :

```
sqlite3

CREATE TABLE shopping (Product text, Quantity int,

               primary key(Product));
```

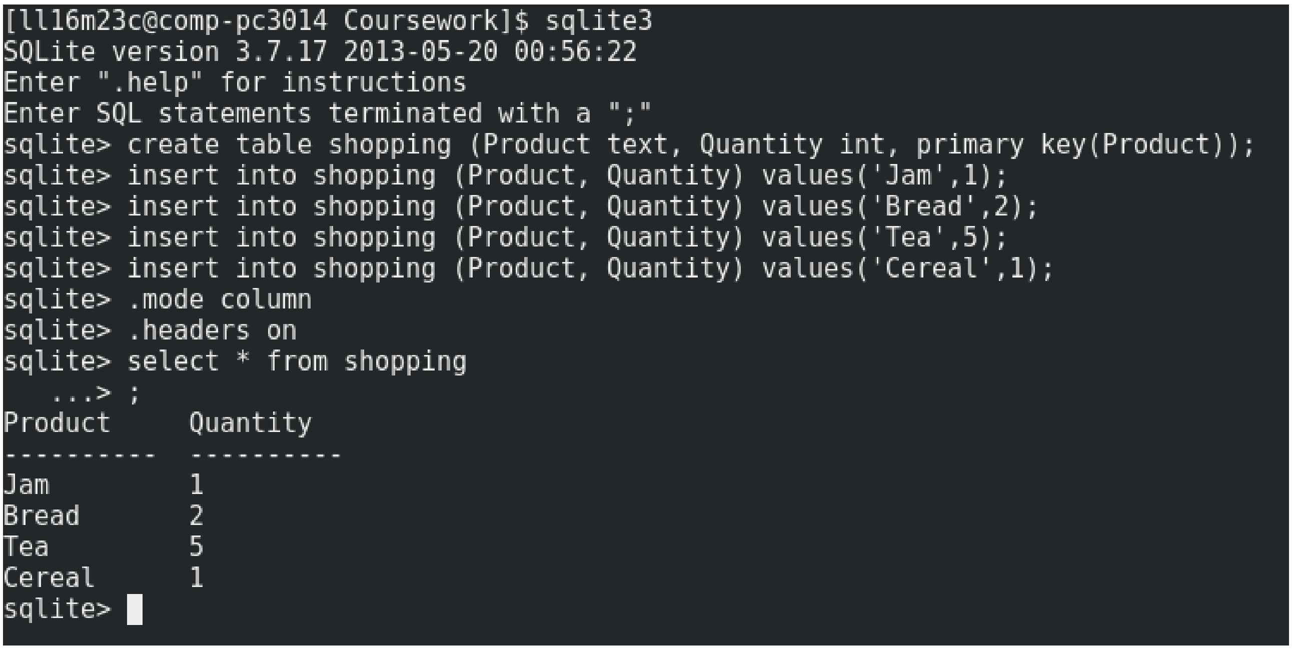
Then to add the data into this table I did the following commands:

```
INSERT INTO shopping (Product, Quantity) VALUES ('JAM',1);

INSERT INTO shopping (Product, Quantity) VALUES ('Bread',2);

INSERT INTO shopping (Product, Quantity) VALUES ('Tea',5);

INSERT INTO shopping (Product, Quantity) VALUES ('Cereal',1);
```

Then to display the data I used these final commands:

```
.mode column

.headers on

SELECT * FROM shopping;
```

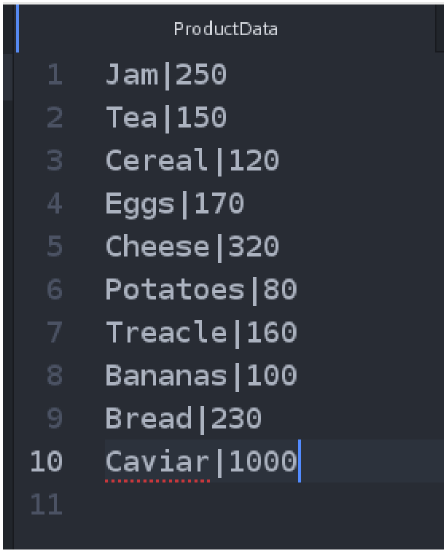
Here is a screen shot of the terminal after all the previous commands:



```
[ll16m23c@comp-pc3014 Coursework]$ sqlite3
SQLite version 3.7.17 2013-05-20 00:56:22
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> create table shopping (Product text, Quantity int, primary key(Product));
sqlite> insert into shopping (Product, Quantity) values('Jam',1);
sqlite> insert into shopping (Product, Quantity) values('Bread',2);
sqlite> insert into shopping (Product, Quantity) values('Tea',5);
sqlite> insert into shopping (Product, Quantity) values('Cereal',1);
sqlite> .mode column
sqlite> .headers on
sqlite> select * from shopping
   ...> ;
Product     Quantity
----------  ----------
Jam         1
Bread       2
Tea         5
Cereal      1
sqlite> 
```

Question 2

I first created the text ProductData in Atom which is shown below:



```
Jam|250
Tea|150
Cereal|120
Eggs|170
Cheese|320
Potatoes|80
Treacle|160
Bananas|100
Bread|230
Caviar|1000
```

I then created the table products using these commands:

```
CREATE TABLE products (Product text, Price int, primary
               key(Product));
```

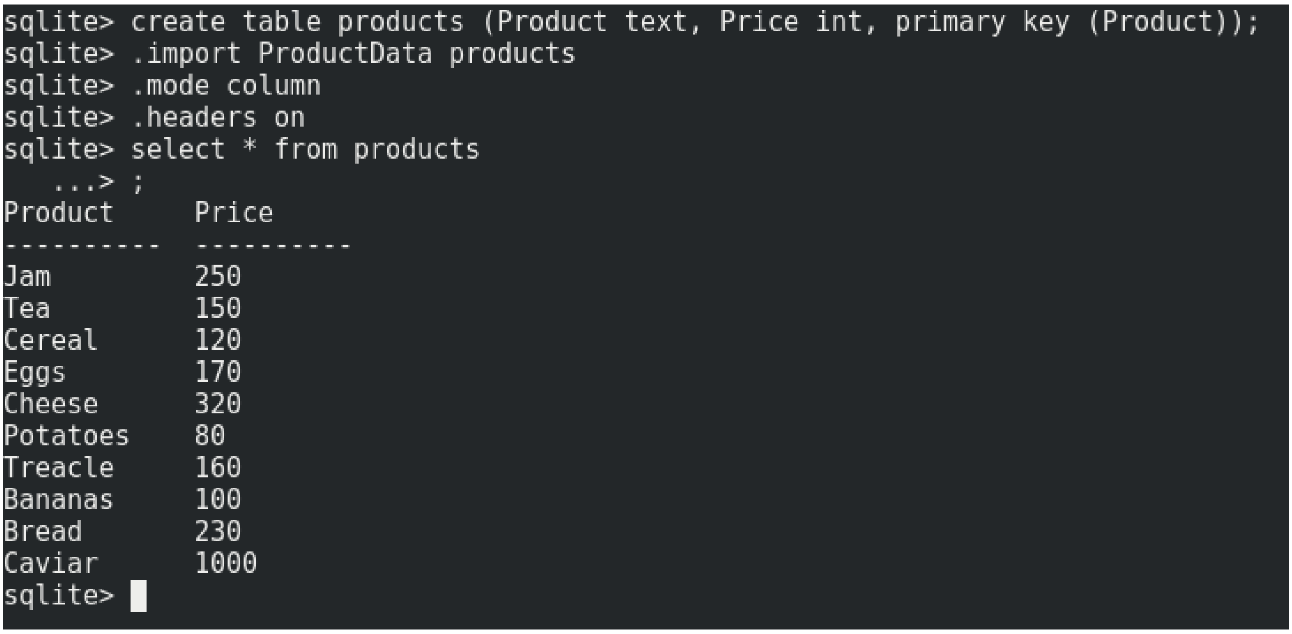
I then imported the data from the text file ProductData into the table with the following command:

```
.import ProductData products
```

I then entered these commands to display the table to check the data was imported correctly:

```
.mode column

.headers on

SELECT * FROM products;
```

Here is a screen shot of the terminal after all of these commands are entered:



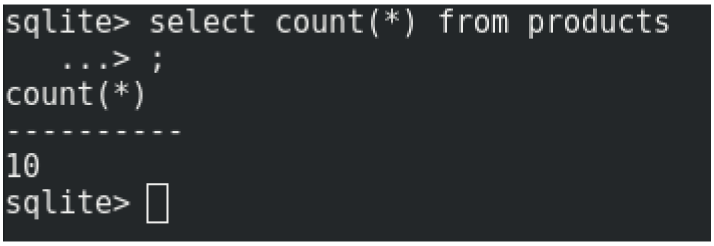
```
sqlite> create table products (Product text, Price int, primary key (Product));
sqlite> .import ProductData products
sqlite> .mode column
sqlite> .headers on
sqlite> select * from products
   ...> ;
Product     Price
----------  ----------
Jam         250
Tea         150
Cereal      120
Eggs        170
Cheese      320
Potatoes    80
Treacle     160
Bananas     100
Bread       230
Caviar      1000
sqlite> 
```

Question 3

In order to find out how many products there are in the table products, a count query should be used on the table products that counts how many rows the table has. The correct answer should be ten. This is the query to find how many products there are:

```
SELECT COUNT (*) FROM products;
```

Here is a screen shot of the terminal after this command is entered:



```
sqlite> select count(*) from products
   ...> ;
count(*)
----------
10
sqlite> 
```

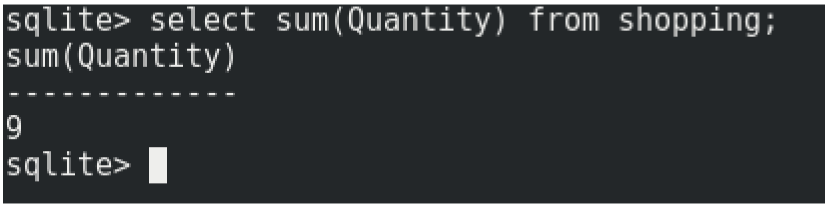
After entering the query, the correct output was shown.

Question 4

In order to find how many items were bought on the shopping trip, the sum query should be used on the table shopping which sums together all quantity entries. The correct answer should be nine. This is the query to find how many items of shopping were bought on the shopping trip:

```
SELECT SUM (quantity) FROM shopping;
```

Here is a screen shot of the terminal after this command is entered:



```
sqlite> select sum(Quantity) from shopping;
sum(Quantity)
-------------
9
sqlite> 
```

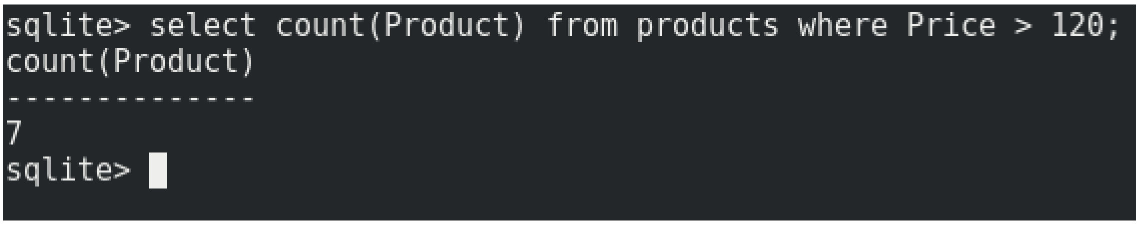
After entering the query, the correct output was shown.

Question 5

To find how many products cost more than 120 the a count query should be used on the products table with the addition of where in the query. The correct output should be seven. This is the query to find how many products cost more than 120:

```
SELECT COUNT (Product) FROM products WHERE Price > 120;
```

Here is a screen shot of the terminal after this command is entered:

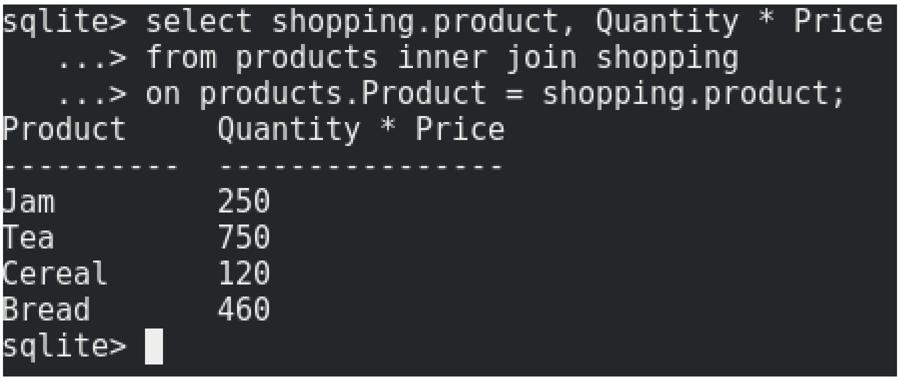


```
sqlite> select count(Product) from products where Price > 120;
count(Product)
--------------
7
sqlite> 
```

After entering the query, the correct output was shown.

Question 6

This is the output in the terminal after entering the query provided in the question:



```
sqlite> select shopping.product, Quantity * Price
   ...> from products inner join shopping
   ...> on products.Product = shopping.product;
Product     Quantity * Price
----------  ----------------
Jam         250
Tea         750
Cereal      120
Bread       460
sqlite> 
```

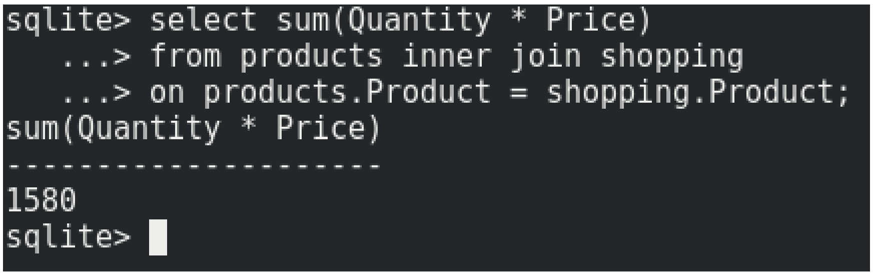
The query is producing how much the shopper spent on each item they bought. So they bought 1 jam at a price of 250 so 1 * 250 = 250, they also bought 5 Tea's at a price of 150 so 5 * 150 = 750 and so on. The table shows the product name and the amount spent on that product.

Question 7

To find the total cost of the shopping the following query should be entered:

```
SELECT SUM (Quantity * Price) FROM products INNER JOIN shopping

          ON products.Product = shopping.Product;
```

The correct output should be 1580. This is the terminal output after entering the query:

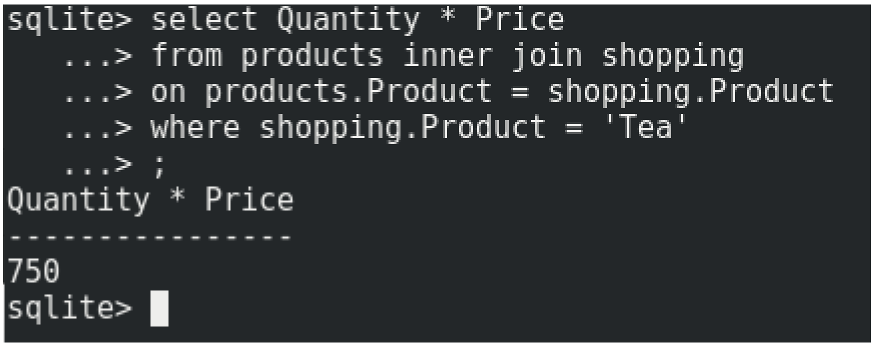


```
sqlite> select sum(Quantity * Price)
   ...> from products inner join shopping
   ...> on products.Product = shopping.Product;
sum(Quantity * Price)
---------------------
1580
sqlite> 
```

As shown, the correct output was produced.

Question 8

This is the output in the terminal after entering the query provided in the question:



```
sqlite> select Quantity * Price
   ...> from products inner join shopping
   ...> on products.Product = shopping.Product
   ...> where shopping.Product = 'Tea'
   ...> ;
Quantity * Price
----------------
750
sqlite> 
```

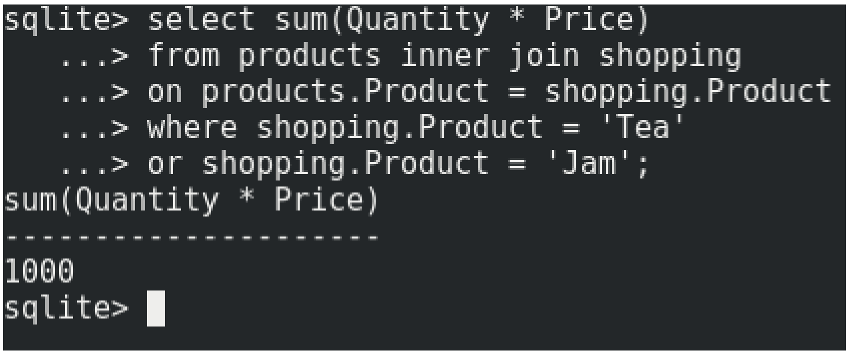
This query is calculating how much the shopper spent on the product Tea. The query is joining together Product data which appears both in the shopping table and products table. However it is also specify to only join the two tables where the data entry Tea is in the shopping table. This only appears once. It is then multiplying the quantity of Tea by the Price of tea and producing this as the output.

Question 9

To answer the question “How much was spent on tea and jam on the shopping trip?”, The following query should be used:

```
SELECT SUM (Quantity * Price) FROM products INNER JOIN shopping
          ON products.Product = shopping.Product WHERE
          shopping.Product = 'Tea' OR shopping.Product = 'Jam';
```

The correct output should be 1000. This is the terminal output after entering the query:



```
sqlite> select sum(Quantity * Price)
   ...> from products inner join shopping
   ...> on products.Product = shopping.Product
   ...> where shopping.Product = 'Tea'
   ...> or shopping.Product = 'Jam';
sum(Quantity * Price)
---------------------
1000
sqlite> 
```

This shows the correct output was produced after entering the query.

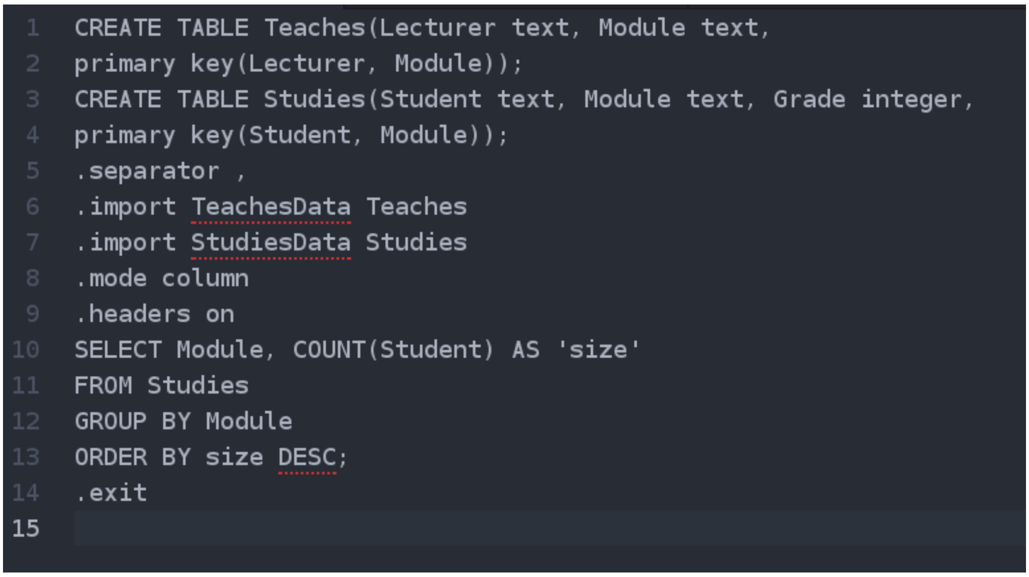
PART B

Initialisation

To answer these questions I created files which executed sql commands and queries to save time and avoid creating the initial tables for every question.

Question 10

To list the modules with the number of students studying that module the following file was used:

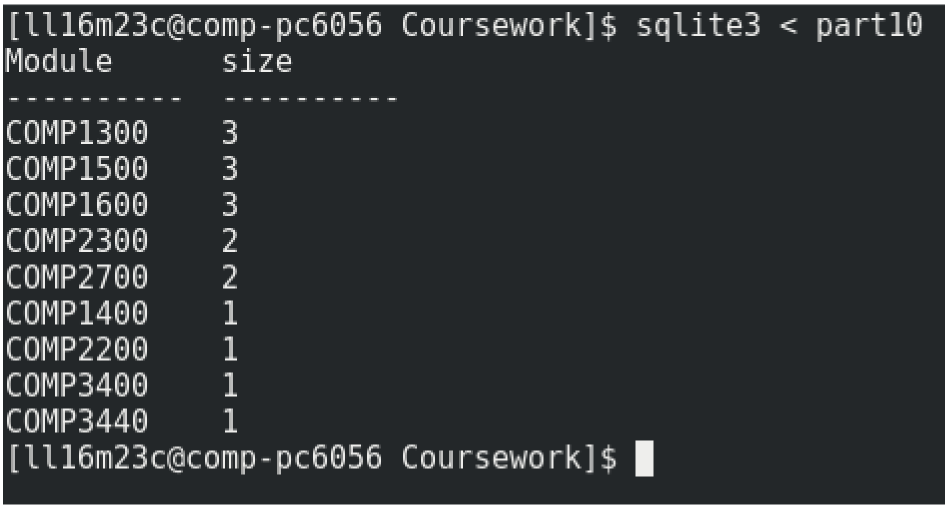


```
CREATE TABLE Teaches(Lecturer text, Module text,
primary key(Lecturer, Module));
CREATE TABLE Studies(Student text, Module text, Grade integer,
primary key(Student, Module));
.separator ,
.import TeachesData Teaches
.import StudiesData Studies
.mode column
.headers on
SELECT Module, COUNT(Student) AS 'size'
FROM Studies
GROUP BY Module
ORDER BY size DESC;
.exit
```

The query used to answer the question was:

```
SELECT Module, COUNT(Student) AS 'size' FROM Studies GROUP BY
               Module ORDER BY size DESC;
```

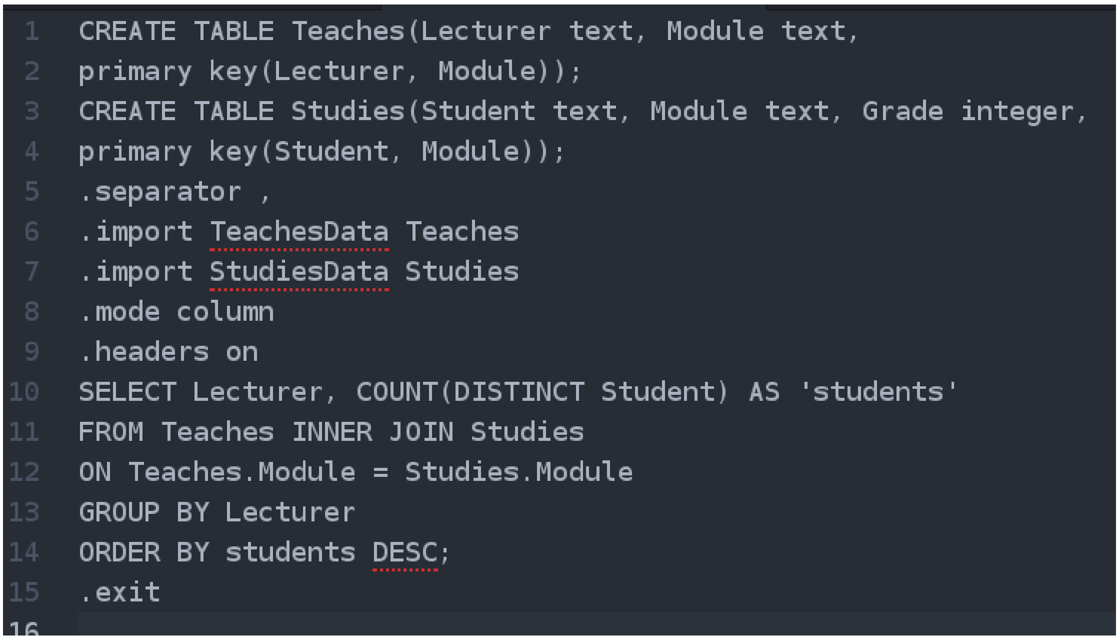
This is the terminal output when the file was executed using sql:



```
[ll16m23c@comp-pc6056 Coursework]$ sqlite3 < part10
Module      size
----------  ----------
COMP1300    3
COMP1500    3
COMP1600    3
COMP2300    2
COMP2700    2
COMP1400    1
COMP2200    1
COMP3400    1
COMP3440    1
[ll16m23c@comp-pc6056 Coursework]$ 
```

Question 11

To list the lecturers with the number of students the lecturer teaches the following file was used:

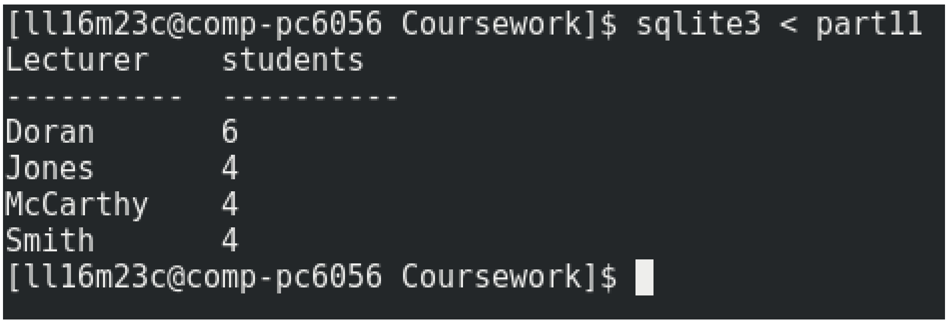


```
CREATE TABLE Teaches(Lecturer text, Module text,
primary key(Lecturer, Module));
CREATE TABLE Studies(Student text, Module text, Grade integer,
primary key(Student, Module));
.separator ,
.import TeachesData Teaches
.import StudiesData Studies
.mode column
.headers on
SELECT Lecturer, COUNT(DISTINCT Student) AS 'students'
FROM Teaches INNER JOIN Studies
ON Teaches.Module = Studies.Module
GROUP BY Lecturer
ORDER BY students DESC;
.exit
```

The query used here is:

```
SELECT Lecturer, COUNT(DISTINCT Student) AS 'students' FROM
     Teaches INNER JOIN Studies ON Teaches.Module = Studies.Module
     GROUP BY Lecturer ORDER BY students DESC;
```

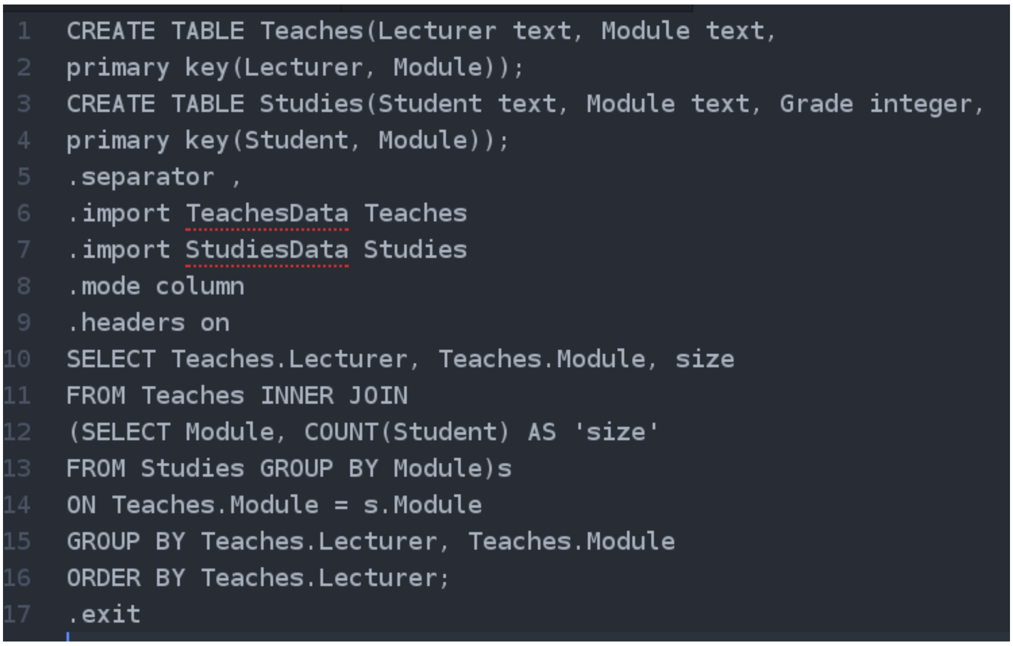
This is the terminal output when the file was executed using sql:



```
[ll16m23c@comp-pc6056 Coursework]$ sqlite3 < part11
Lecturer    students
----------  ----------
Doran       6
Jones       4
McCarthy    4
Smith       4
[ll16m23c@comp-pc6056 Coursework]$ 
```

Question 12

To list each lecturer with each module they teach and the number of students studying that module the following file was used:

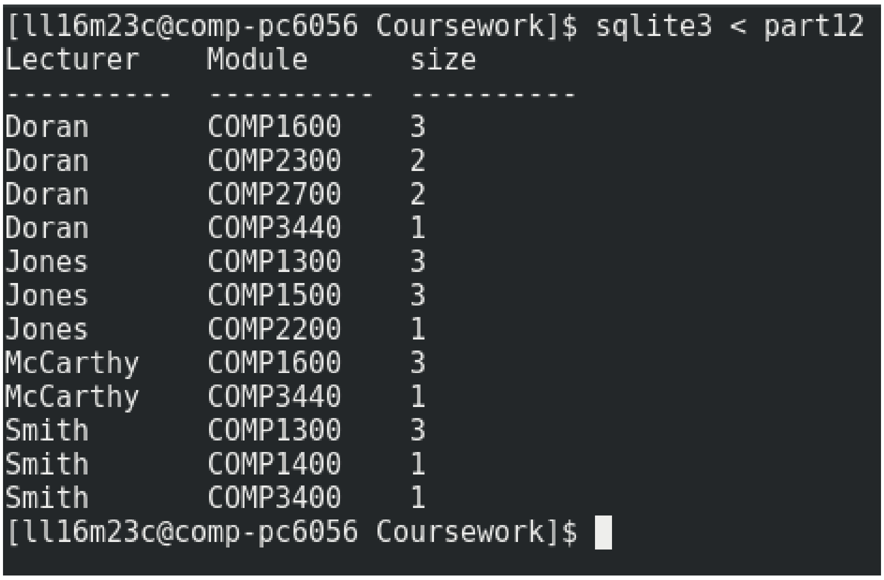


```
CREATE TABLE Teaches(Lecturer text, Module text,
primary key(Lecturer, Module));
CREATE TABLE Studies(Student text, Module text, Grade integer,
primary key(Student, Module));
.separator ,
.import TeachesData Teaches
.import StudiesData Studies
.mode column
.headers on
SELECT Teaches.Lecturer, Teaches.Module, size
FROM Teaches INNER JOIN
(SELECT Module, COUNT(Student) AS 'size'
FROM Studies GROUP BY Module)s
ON Teaches.Module = s.Module
GROUP BY Teaches.Lecturer, Teaches.Module
ORDER BY Teaches.Lecturer;
.exit
```

The queries used here are:

```
SELECT Teaches.Lecturer, Teaches.Module, size FROM Teaches INNER
     JOIN
     (SELECT Module, COUNT(Student) AS 'size' FROM Studies
          GROUP BY Module)s
     ON Teaches.Module = s.Module GROUP BY  Teaches.Lecturer,
     Teaches.Module ORDER BY Teaches.Lecturer;
```

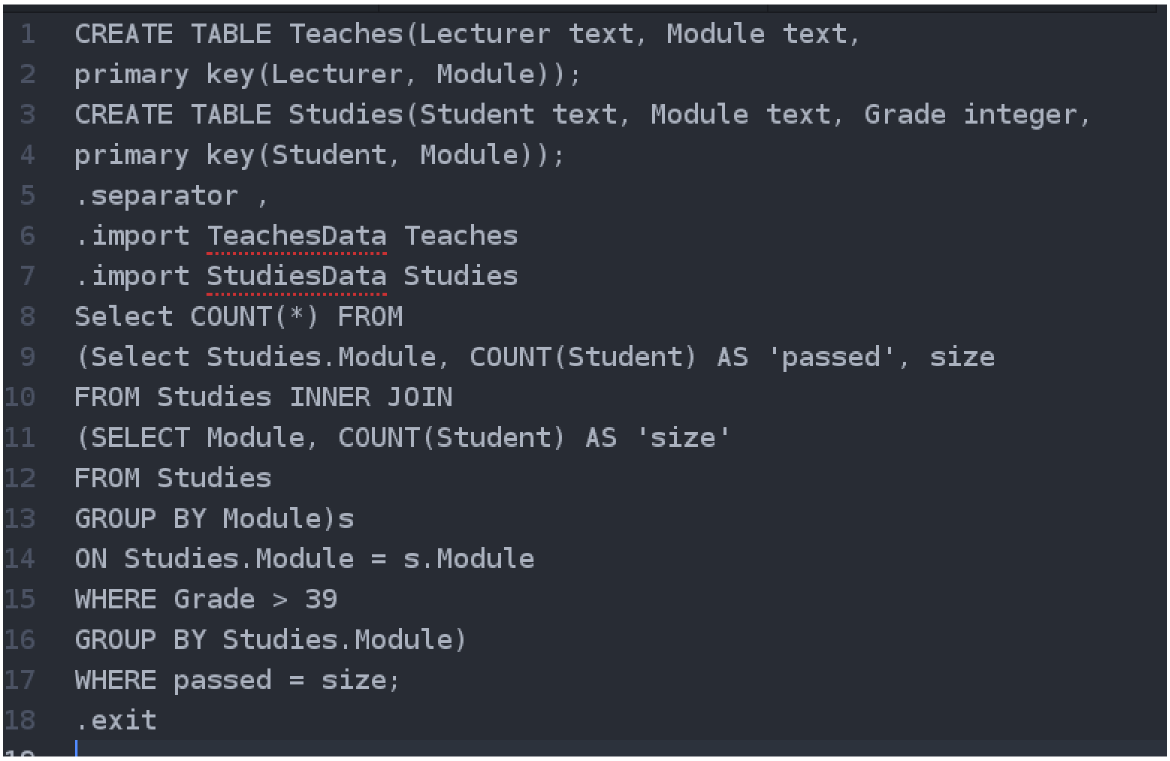
This is the terminal output after executing the file using sql:



```
[ll16m23c@comp-pc6056 Coursework]$ sqlite3 < part12
Lecturer    Module      size
----------  ----------  ----------
Doran       COMP1600    3
Doran       COMP2300    2
Doran       COMP2700    2
Doran       COMP3440    1
Jones       COMP1300    3
Jones       COMP1500    3
Jones       COMP2200    1
McCarthy    COMP1600    3
McCarthy    COMP3440    1
Smith       COMP1300    3
Smith       COMP1400    1
Smith       COMP3400    1
[ll16m23c@comp-pc6056 Coursework]$ 
```

Question 13

To find the number of modules in which everyone passed the module the following file was used:

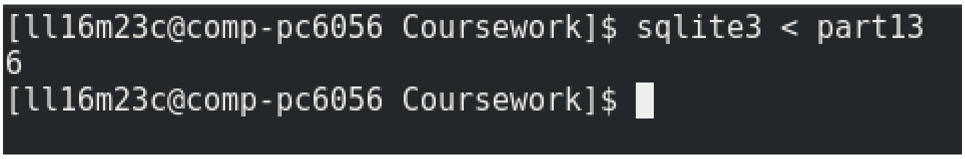


```
CREATE TABLE Teaches(Lecturer text, Module text,
primary key(Lecturer, Module));
CREATE TABLE Studies(Student text, Module text, Grade integer,
primary key(Student, Module));
.separator ,
.import TeachesData Teaches
.import StudiesData Studies
Select COUNT(*) FROM
(Select Studies.Module, COUNT(Student) AS 'passed', size
FROM Studies INNER JOIN
(SELECT Module, COUNT(Student) AS 'size'
FROM Studies
GROUP BY Module)s
ON Studies.Module = s.Module
WHERE Grade > 39
GROUP BY Studies.Module)
WHERE passed = size;
.exit
```

The queries used here are:

```
Select COUNT(*) FROM

     (Select Studies.Module, COUNT(Student) AS
          'passed', size FROM Studies INNER JOIN

          (SELECT Module, COUNT(Student) AS 'size' FROM Studies
               GROUP BY Module)s

          ON Studies.Module = s.Module WHERE Grade > 39 GROUP BY
          Studies.Module)

     WHERE passed = size;
```

This is the terminal output after executing the file using sql:



```
[ll16m23c@comp-pc6056 Coursework]$ sqlite3 < part13
6
[ll16m23c@comp-pc6056 Coursework]$ 
```

There are 6 Modules in which everyone that studies the module passed the module.