

# PART A

## Question 1

To create Table 1 I used the following commands :

```
sqlite3
```

```
CREATE TABLE shopping (Product text, Quantity int,  
                        primary key(Product));
```

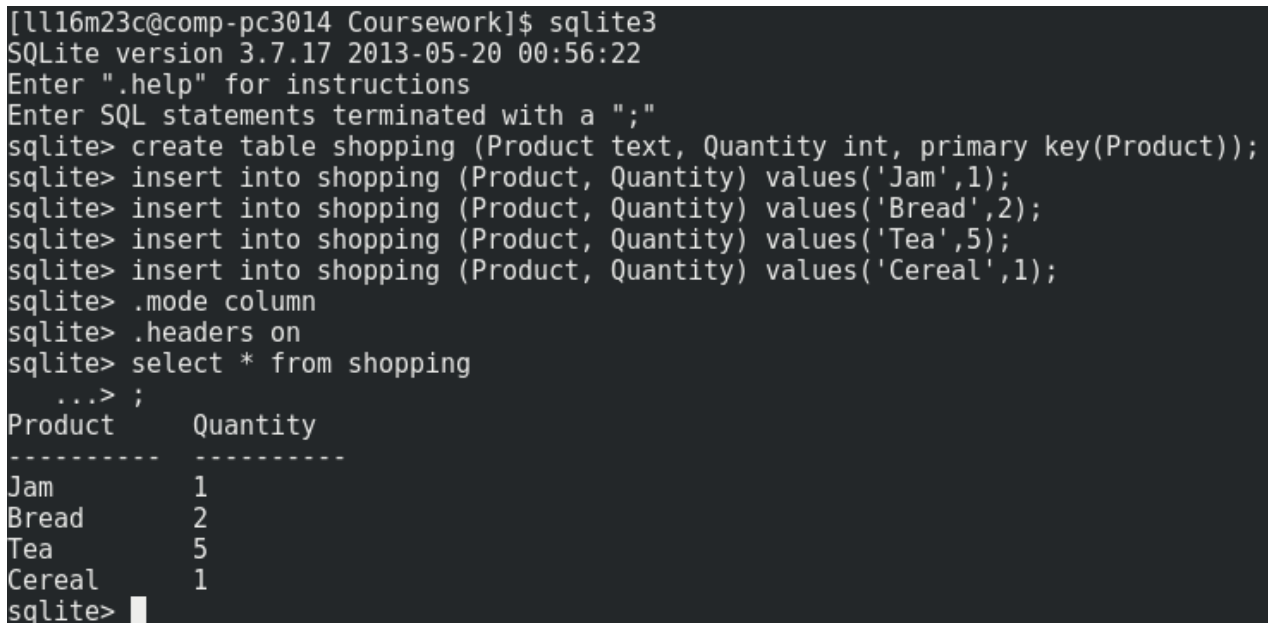
Then to add the data into this table I did the following commands:

```
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:

A screenshot of the Atom text editor showing a file named 'ProductData'. The file contains a list of 10 items, each on a new line, separated by a vertical bar. The items are: 1 Jam|250, 2 Tea|150, 3 Cereal|120, 4 Eggs|170, 5 Cheese|320, 6 Potatoes|80, 7 Treacle|160, 8 Bananas|100, 9 Bread|230, and 10 Caviar|1000. The cursor is positioned at the end of the 10th line.

```
ProductData
1 Jam|250
2 Tea|150
3 Cereal|120
4 Eggs|170
5 Cheese|320
6 Potatoes|80
7 Treacle|160
8 Bananas|100
9 Bread|230
10 Caviar|1000
11
```

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
```

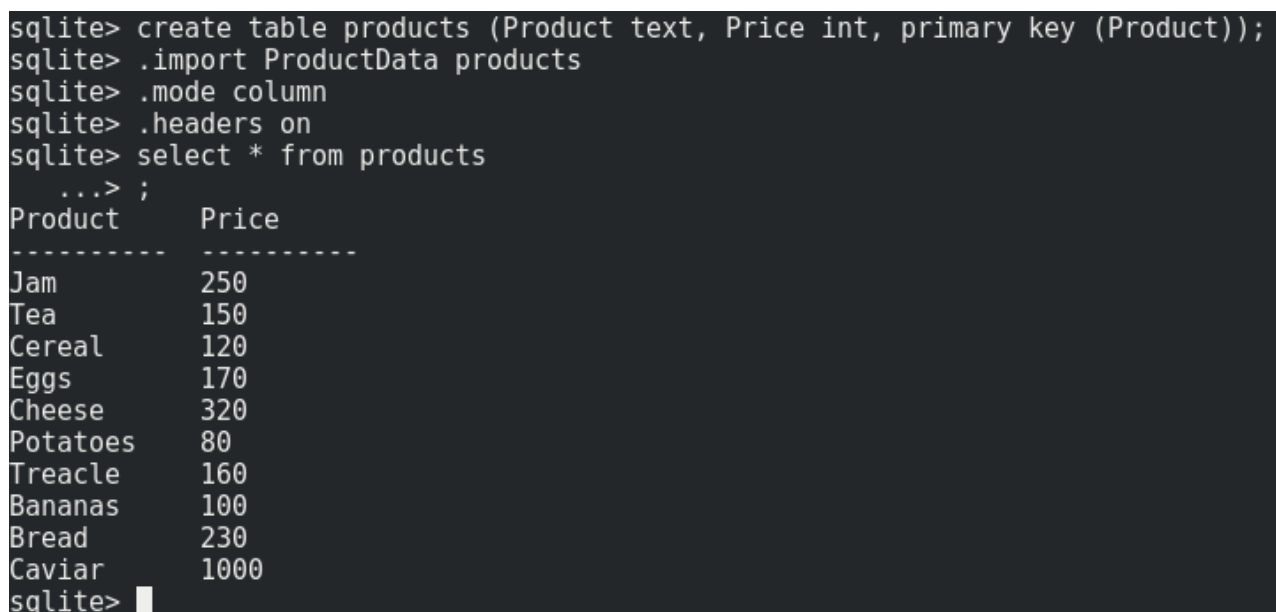
I then entered these commands to display the table to check the data was imported correctly:

```
.mode column
```

```
.headers on
```

```
SELECT * FROM products;
```

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

A screenshot of a terminal window showing the execution of SQLite commands. The commands are: create table products (Product text, Price int, primary key (Product));, .import ProductData products, .mode column, .headers on, and select \* from products. The output shows a table with two columns: Product and Price. The data is as follows:

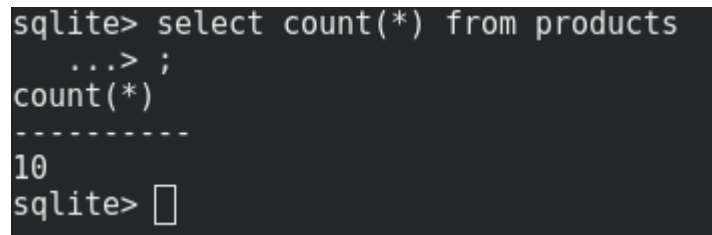
```
sqlite> create table products (Product text, Price int, primary key (Product));
sqlite> .import ProductData products
sqlite> .mode column
sqlite> .headers on
sqlite> select * from products
...> ;
Product      Price
-----
Jam           250
Tea           150
Cereal        120
Eggs          170
Cheese        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:

A terminal window with a dark background. The prompt 'sqlite>' is followed by the command 'select count(\*) from products'. The command is executed, and the output 'count(\*)' is shown with a dashed line separator below it, followed by the value '10'. The prompt 'sqlite>' is shown again with a cursor.

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

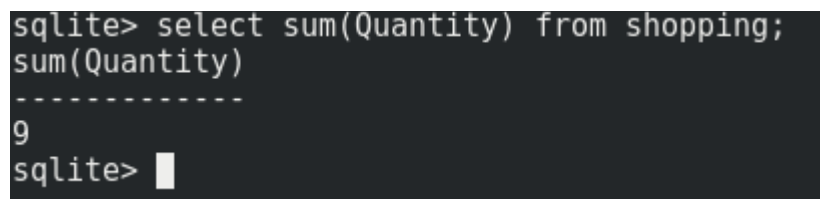
After entering the query, the correct output was shown.

## 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:

A terminal window with a dark background. The prompt 'sqlite>' is followed by the command 'select sum(Quantity) from shopping;'. The command is executed, and the output 'sum(Quantity)' is shown with a dashed line separator below it, followed by the value '9'. The prompt 'sqlite>' is shown again with a cursor.

```
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> █
```

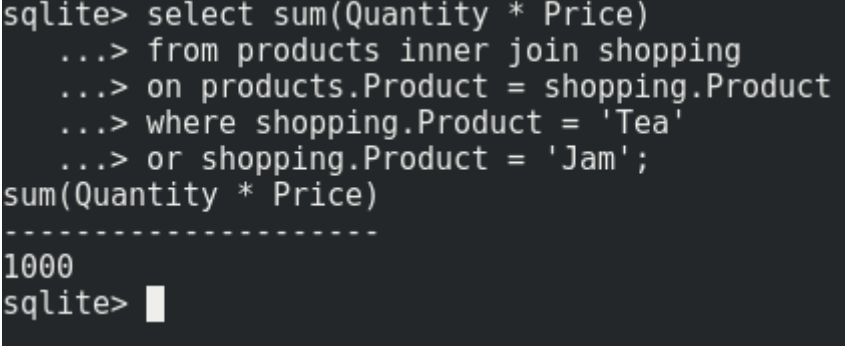
This query is calculating how much the shopper spent on the product Tea. The query is joining together Product data which appears both in the shopping table and products table. However it is also specify to only join the two tables where the data entry Tea is in the shopping table. This only appears 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:

A terminal window with a dark background and light-colored text. The text shows a SQLite command prompt where a SQL query is entered and executed. The query calculates the sum of (Quantity \* Price) for products 'Tea' and 'Jam'. The output shows the column name 'sum(Quantity \* Price)' followed by a dashed line and the value '1000'. The prompt returns to 'sqlite>' with a cursor.

```
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:

```
1 CREATE TABLE Teaches(Lecturer text, Module text,  
2 primary key(Lecturer, Module));  
3 CREATE TABLE Studies(Student text, Module text, Grade integer,  
4 primary key(Student, Module));  
5 .separator ,  
6 .import TeachesData Teaches  
7 .import StudiesData Studies  
8 .mode column  
9 .headers on  
10 SELECT Module, COUNT(Student) AS 'size'  
11 FROM Studies  
12 GROUP BY Module  
13 ORDER BY size DESC;  
14 .exit  
15
```

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:

```
1 CREATE TABLE Teaches(Lecturer text, Module text,  
2 primary key(Lecturer, Module));  
3 CREATE TABLE Studies(Student text, Module text, Grade integer,  
4 primary key(Student, Module));  
5 .separator ,  
6 .import TeachesData Teaches  
7 .import StudiesData Studies  
8 .mode column  
9 .headers on  
10 SELECT Lecturer, COUNT(DISTINCT Student) AS 'students'  
11 FROM Teaches INNER JOIN Studies  
12 ON Teaches.Module = Studies.Module  
13 GROUP BY Lecturer  
14 ORDER BY students DESC;  
15 .exit  
16
```

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:

```
1 CREATE TABLE Teaches(Lecturer text, Module text,  
2 primary key(Lecturer, Module));  
3 CREATE TABLE Studies(Student text, Module text, Grade integer,  
4 primary key(Student, Module));  
5 .separator ,  
6 .import TeachesData Teaches  
7 .import StudiesData Studies  
8 .mode column  
9 .headers on  
10 SELECT Teaches.Lecturer, Teaches.Module, size  
11 FROM Teaches INNER JOIN  
12 (SELECT Module, COUNT(Student) AS 'size'  
13 FROM Studies GROUP BY Module)s  
14 ON Teaches.Module = s.Module  
15 GROUP BY Teaches.Lecturer, Teaches.Module  
16 ORDER BY Teaches.Lecturer;  
17 .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:

```
[lll16m23c@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  
[lll16m23c@comp-pc6056 Coursework]$
```

## Question 13

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

```
1 CREATE TABLE Teaches(Lecturer text, Module text,  
2 primary key(Lecturer, Module));  
3 CREATE TABLE Studies(Student text, Module text, Grade integer,  
4 primary key(Student, Module));  
5 .separator ,  
6 .import TeachesData Teaches  
7 .import StudiesData Studies  
8 Select COUNT(*) FROM  
9 (Select Studies.Module, COUNT(Student) AS 'passed', size  
10 FROM Studies INNER JOIN  
11 (SELECT Module, COUNT(Student) AS 'size'  
12 FROM Studies  
13 GROUP BY Module)s  
14 ON Studies.Module = s.Module  
15 WHERE Grade > 39  
16 GROUP BY Studies.Module)  
17 WHERE passed = size;  
18 .exit  
19
```

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.