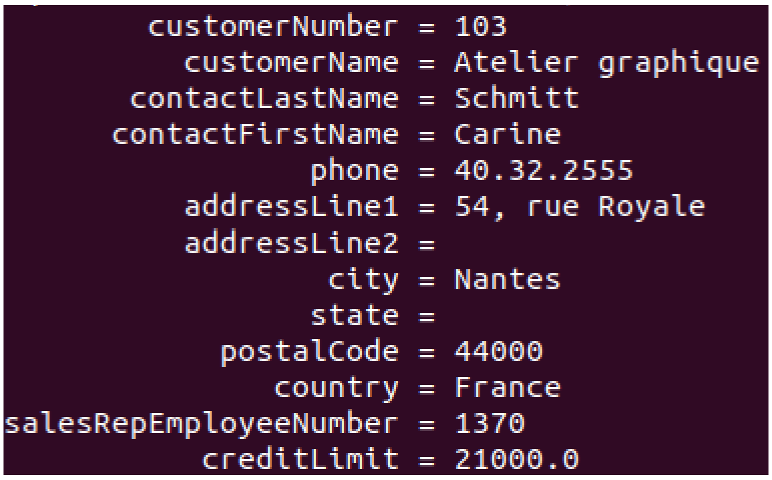
# INF-2700
# Assignment 1
# Magnus Dahl-Hansen

## Part1, Task 1:

***Describe the inf2700_orders database schema.***
***It is sufficient to show the CREATE TABLE statements that defined the schema.***

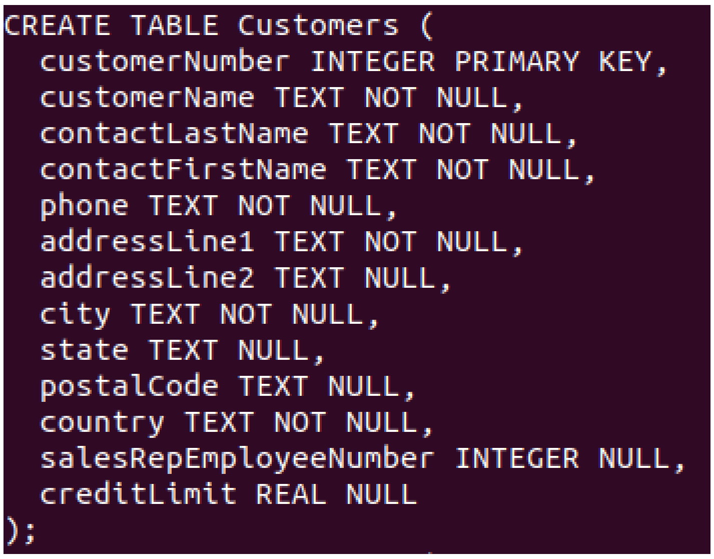
The database schema “inf2700_orders.sqlite3” consists of different tables created by the “CREATE TABLE” command. The tables which the schema consists of are: “Customers”, “Employes”, “Offices”, “OrderDetails”, “Orders”, “Payments”, “Products” and “ProductLines”.

The different tables again consists of different rows and columns that contain useful information. For example does the “Customers” table contain the following information about the customer Atelier Graphique:



```
        customerNumber = 103
          customerName = Atelier graphique
       contactLastName = Schmitt
      contactFirstName = Carine
                 phone = 40.32.2555
          addressLine1 = 54, rue Royale
          addressLine2 =
                  city = Nantes
                 state =
            postalCode = 44000
               country = France
salesRepEmployeeNumber = 1370
           creditLimit = 21000.0
```

It is specified what the different tables will consist of the the tables are created. The picture below describes which columns are going to be created and of what type they are going to be. As an example is it specified that the column “CustomerNumber” will be of type *integer* and will be a *PRIMARY KEY.*

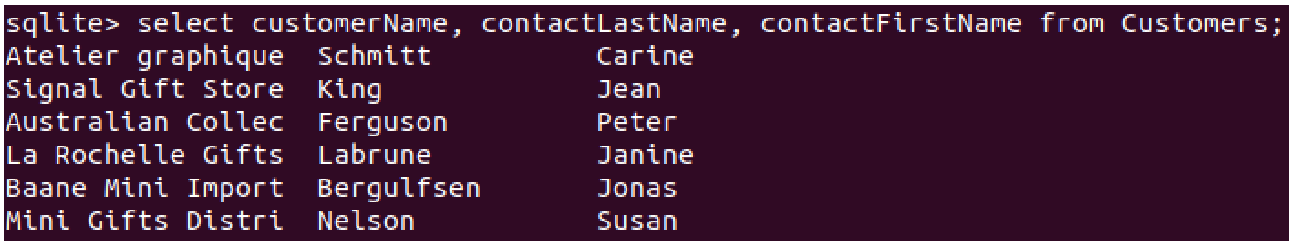


```
CREATE TABLE Customers (
  customerNumber INTEGER PRIMARY KEY,
  customerName TEXT NOT NULL,
  contactLastName TEXT NOT NULL,
  contactFirstName TEXT NOT NULL,
  phone TEXT NOT NULL,
  addressLine1 TEXT NOT NULL,
  addressLine2 TEXT NULL,
  city TEXT NOT NULL,
  state TEXT NULL,
  postalCode TEXT NULL,
  country TEXT NOT NULL,
  salesRepEmployeeNumber INTEGER NULL,
  creditLimit REAL NULL
);
```

## Part1, Task2:

***Run the queries listed below. For each query, write in your own words in a few lines that describe its purpose and how it works.***

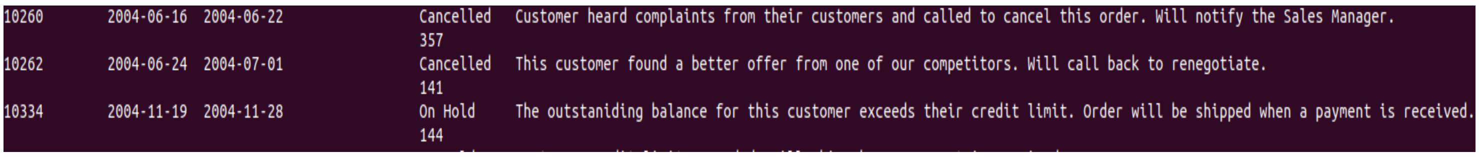
**Task 2, a)**
***SELECT*** *customerName, contactLastName, contactFirstName*
***FROM*** *Customers;*



```
sqlite> select customerName, contactLastName, contactFirstName from Customers;
Atelier graphique  Schmitt          Carine
Signal Gift Store  King             Jean
Australian Collec  Ferguson         Peter
La Rochelle Gifts  Labrune          Janine
Baane Mini Import  Bergulfsen       Jonas
Mini Gifts Distri  Nelson           Susan
```

This query extracts and displays the customers name, last name and first name from the table "Customers".

**Task 2, b)**
***SELECT * FROM*** *Orders* ***WHERE*** *shippedDate* ***IS NULL****;*

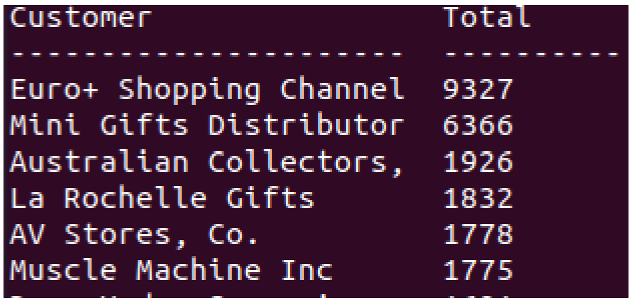


```
10260        2004-06-16  2004-06-22                     Cancelled   Customer heard complaints from their customers and called to cancel this order. Will notify the Sales Manager.
                                                        357
10262        2004-06-24  2004-07-01                     Cancelled   This customer found a better offer from one of our competitors. Will call back to renegotiate.
                                                        141
10334        2004-11-19  2004-11-28                     On Hold     The outstaniding balance for this customer exceeds their credit limit. Order will be shipped when a payment is received.
                                                        144
```

This query gives information about which shipments are not complete. The query is performed in these steps:

1. Select all information from the table Orders.
2. Single out and display only the shipments that is NULL/not complete

**c)** *SELECT C.customerName AS Customer, SUM(OD.quantityOrdered) AS Total FROM Orders O, Customers C, OrderDetails OD WHERE O.customerNumber = C.customerNumber AND O.orderNumber = OD.orderNumber GROUP BY O.customerNumber ORDER BY Total DESC;*



```
Customer                 Total
----------------------   ----------
Euro+ Shopping Channel   9327
Mini Gifts Distributor   6366
Australian Collectors,   1926
La Rochelle Gifts        1832
AV Stores, Co.           1778
Muscle Machine Inc       1775
```

The query gives information about how many orders each customer has made and is performed in these steps:

1. Fetch customerName rows from the Customer table and display them under a column named "Customer"
2. Fetch quantityOrdered rows from the OrderDetails table, sum up all the orders for each customer and display them under the column named "Total".

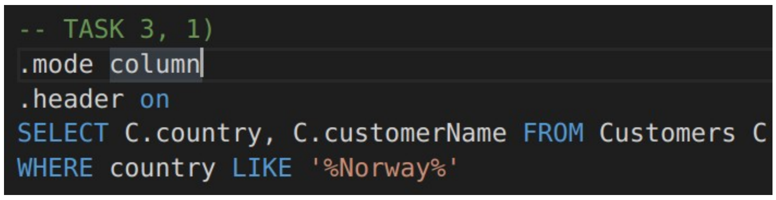
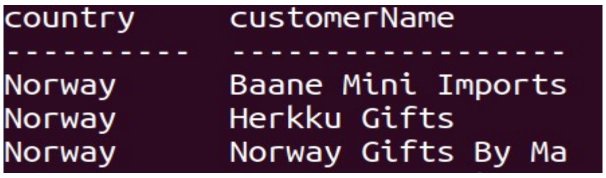
3. Specify that "C." relates to the table Customers, "OD." relates to the OrderDetails table and that "O." relates to the Orders table.
4. Fetch only the "customerNumber" rows that match from the two tables Customers and Orders.
5. Fetch only the "orderNumber" rows that match from the two tables Orders and OrderDetails.
6. Group the search results by the customers numbers
7. Display the given search results by "orderQuantity" in a descending order.

**Task 2, d)**
***SELECT*** *P.productName, T.totalQuantityOrdered*
***FROM*** *Products P* ***NATURAL JOIN (SELECT*** *productCode,* ***SUM(****quantityOrdered****) AS*** *totalQuantityOrdered*
***FROM*** *OrderDetails*
***GROUP BY*** *productCode****) AS*** *T*
***WHERE*** *T.totalQuantityOrdered >= 1000;*

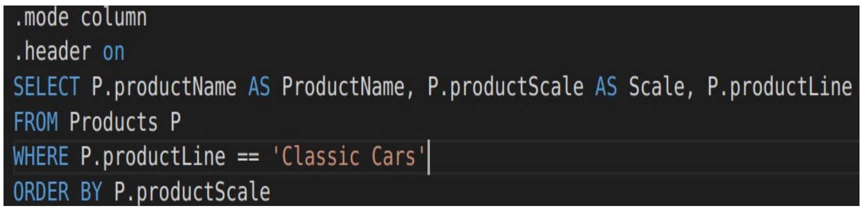
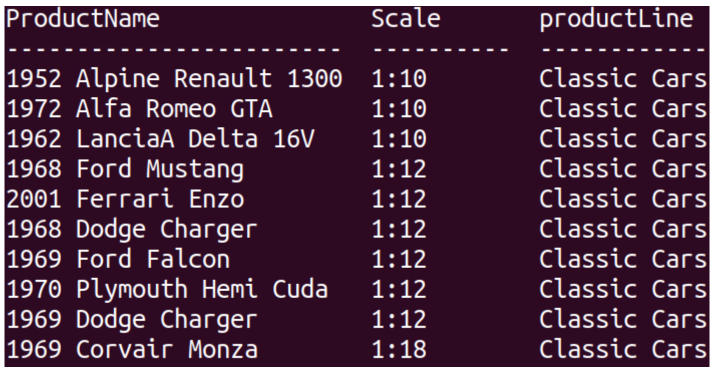
The query gives information about a products name and how many times it has been sold. It will also only display the products that has been sold more than or equal to 1000 times.
The query is performed in these steps:
1. Fetch *productName* row from the *Products* table
2. Join the productCode row from the OrderDetails table with the sum of the quantityOrdered row from the OrderDetails table
3. Fetch productCode row from the OrderDetails table
4. Display only the results that were ordered more than or equal to 1000 times

## Part1, Task 3:
***Solve the following problems in SQL:***

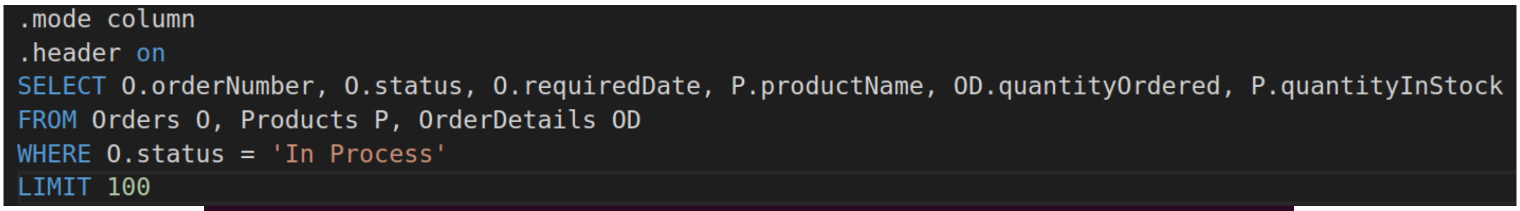
**Task 3, 1)**
***Retrieve all customers in Norway.***


```
-- TASK 3, 1)
.mode column
.header on
SELECT C.country, C.customerName FROM Customers C
WHERE country LIKE '%Norway%'
```

```
country     customerName
----------  ------------------
Norway      Baane Mini Imports
Norway      Herkku Gifts
Norway      Norway Gifts By Ma
```

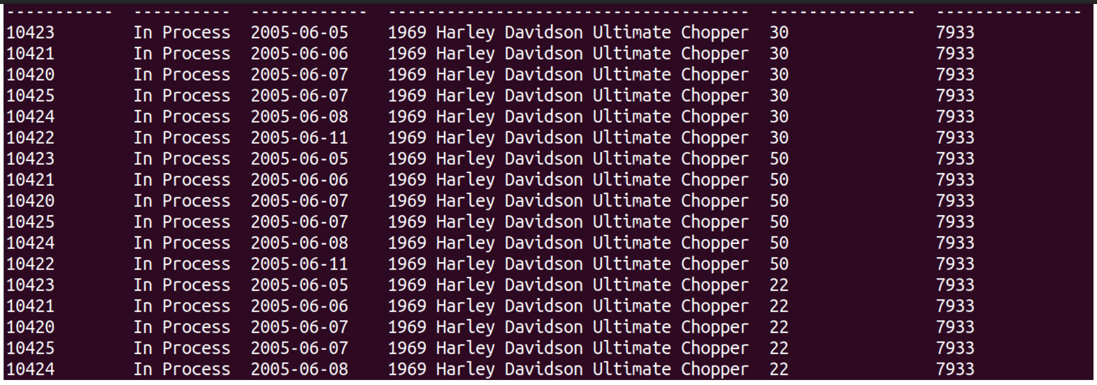
**Task 3, 2)**
***Retrieve all classic car products and their scale.***


```
.mode column
.header on
SELECT P.productName AS ProductName, P.productScale AS Scale, P.productLine
FROM Products P
WHERE P.productLine == 'Classic Cars'
ORDER BY P.productScale
```

```
ProductName               Scale       productLine
------------------------  ----------  ------------
1952 Alpine Renault 1300  1:10        Classic Cars
1972 Alfa Romeo GTA       1:10        Classic Cars
1962 LanciaA Delta 16V    1:10        Classic Cars
1968 Ford Mustang         1:12        Classic Cars
2001 Ferrari Enzo         1:12        Classic Cars
1968 Dodge Charger        1:12        Classic Cars
1969 Ford Falcon          1:12        Classic Cars
1970 Plymouth Hemi Cuda   1:12        Classic Cars
1969 Dodge Charger        1:12        Classic Cars
1969 Corvair Monza        1:18        Classic Cars
```

**Task 3, 3)**
***Create a list of incomplete orders (order status is "In process"). The list must contain order-Number, requiredDate, productName, quantityOrdered and quantityInStock.***

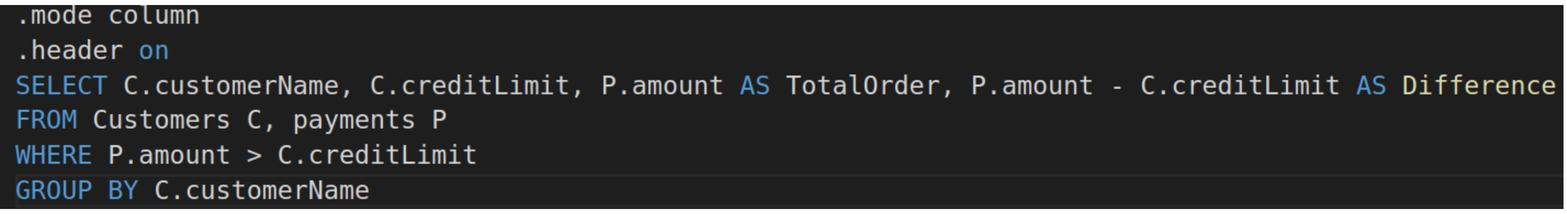


```
.mode column
.header on
SELECT O.orderNumber, O.status, O.requiredDate, P.productName, OD.quantityOrdered, P.quantityInStock
FROM Orders O, Products P, OrderDetails OD
WHERE O.status = 'In Process'
LIMIT 100
```

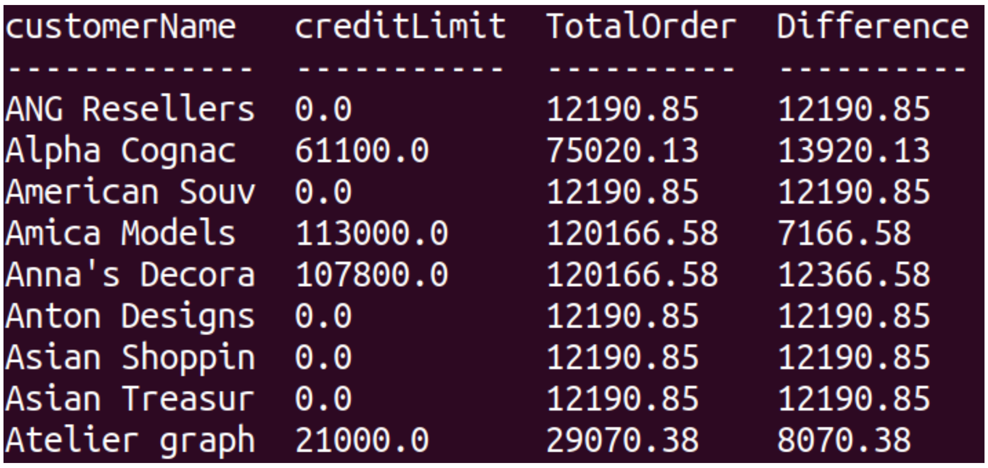


```
----------  ----------  -----------  ------------------------------------  --------------  --------------
10423       In Process  2005-06-05   1969 Harley Davidson Ultimate Chopper  30              7933
10421       In Process  2005-06-06   1969 Harley Davidson Ultimate Chopper  30              7933
10420       In Process  2005-06-07   1969 Harley Davidson Ultimate Chopper  30              7933
10425       In Process  2005-06-07   1969 Harley Davidson Ultimate Chopper  30              7933
10424       In Process  2005-06-08   1969 Harley Davidson Ultimate Chopper  30              7933
10422       In Process  2005-06-11   1969 Harley Davidson Ultimate Chopper  30              7933
10423       In Process  2005-06-05   1969 Harley Davidson Ultimate Chopper  50              7933
10421       In Process  2005-06-06   1969 Harley Davidson Ultimate Chopper  50              7933
10420       In Process  2005-06-07   1969 Harley Davidson Ultimate Chopper  50              7933
10425       In Process  2005-06-07   1969 Harley Davidson Ultimate Chopper  50              7933
10424       In Process  2005-06-08   1969 Harley Davidson Ultimate Chopper  50              7933
10422       In Process  2005-06-11   1969 Harley Davidson Ultimate Chopper  50              7933
10423       In Process  2005-06-05   1969 Harley Davidson Ultimate Chopper  22              7933
10421       In Process  2005-06-06   1969 Harley Davidson Ultimate Chopper  22              7933
10420       In Process  2005-06-07   1969 Harley Davidson Ultimate Chopper  22              7933
10425       In Process  2005-06-07   1969 Harley Davidson Ultimate Chopper  22              7933
10424       In Process  2005-06-08   1969 Harley Davidson Ultimate Chopper  22              7933
```

**Task 3, 4)**
***Create a list of customers where the difference between the total price of all ordered products and the total amount of all payments exceeds the credit limit. The list must contain the customer name, credit limit, total price, total payment and the difference between the two sums.***

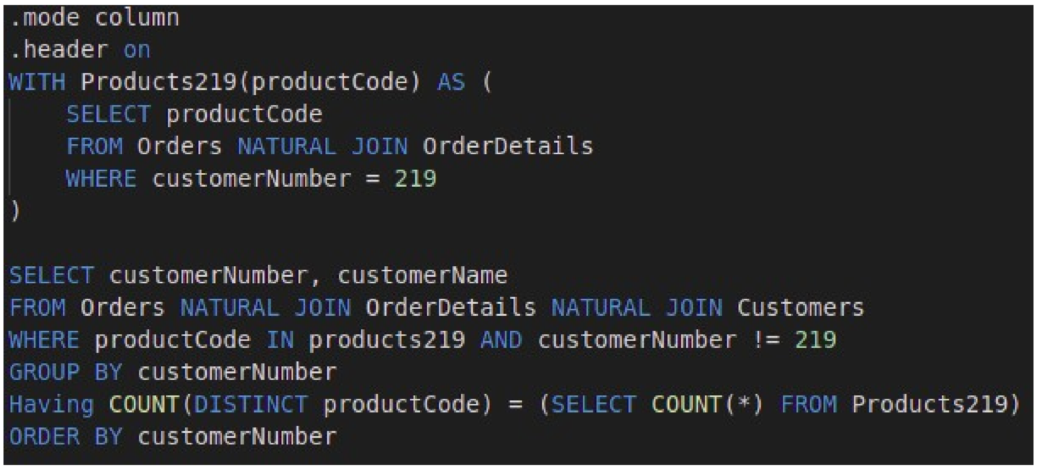


```
.mode column
.header on
SELECT C.customerName, C.creditLimit, P.amount AS TotalOrder, P.amount - C.creditLimit AS Difference
FROM Customers C, payments P
WHERE P.amount > C.creditLimit
GROUP BY C.customerName
```



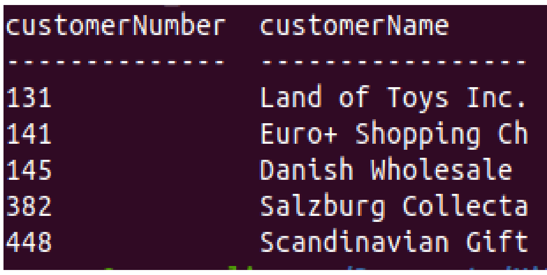
```
customerName   creditLimit  TotalOrder  Difference
-------------  -----------  ----------  ----------
ANG Resellers  0.0          12190.85    12190.85
Alpha Cognac   61100.0      75020.13    13920.13
American Souv  0.0          12190.85    12190.85
Amica Models   113000.0     120166.58   7166.58
Anna's Decora  107800.0     120166.58   12366.58
Anton Designs  0.0          12190.85    12190.85
Asian Shoppin  0.0          12190.85    12190.85
Asian Treasur  0.0          12190.85    12190.85
Atelier graph  21000.0      29070.38    8070.38
```

**Task 3, 5)**
***Create a list of customers who have ordered all products that customer 219 has ordered.***



```
.mode column
.header on
WITH Products219(productCode) AS (
    SELECT productCode
    FROM Orders NATURAL JOIN OrderDetails
    WHERE customerNumber = 219
)

SELECT customerNumber, customerName
FROM Orders NATURAL JOIN OrderDetails NATURAL JOIN Customers
WHERE productCode IN products219 AND customerNumber != 219
GROUP BY customerNumber
Having COUNT(DISTINCT productCode) = (SELECT COUNT(*) FROM Products219)
ORDER BY customerNumber
```



```
customerNumber  customerName
--------------  ----------------
131             Land of Toys Inc.
141             Euro+ Shopping Ch
145             Danish Wholesale
382             Salzburg Collecta
448             Scandinavian Gift
```

## Part2:
***Implement a SQL tester in C.***

The implementation of the SQL tester is solved by using the *<sqlite3.h>* library. The library has built-in functions for opening a given database for further use and for closing the database.

The given query is executed by sending a copy of it to the *sqlite3_exec()* function. The choice to send a copy of the query was made to ensure that no problems with the original query line arose later on. The *sqlite3_exec()* function takes in five arguments which includes a database that is going to be used, a query-line to be executed against the database, a pointer to a callback function and a pointer to the arguments that is passed to the callback function.

The callback function in this implementation is heavily based on the callback function from the book: *“Owens, M. and Hipp, D., 2006. The definitive guide to SQLite. New York: Apress, page 210*”. The function retrieves results from a SELECT statement and places this data in a pointer which later on is printed to the terminal. This select and retrieve process is repeated for every SELECT statement until the end of the query is reached.